mc 8

Die Mikrocomputer-Zeitschrift

7 DM · 60 öS · 7 sfr. · August 1987

Der AT und die 1-MByte-Grenze

Bus-Systeme:

IBM Micro Channel und Apple NuBus

Grundlagen:
Transputer
Monte-Carlo-Verfahren

Software:
GW-Basic ruft MS-DOS
Screen-Saver für Macintosh

Serien:
mc-AllProg, Unix

Test:
386-PC, IBM Modell 60, Betriebssystem QNX

Liebe Leser!

Das Titelbild in diesem Heft hat Streit in die Redaktion gebracht: Wie kann man nur eine intakte AT-Platine hernehmen und für fotografische Zwecke einfach durchbohren!? – so wurde anklagend gerufen. Beim Entwurf des Titelbildes schien die Aussage, daß IBM mit seinem neuen Bus-Konzept auch die gerade laufende AT-Clone-Welle beeinträchtigen wolle, den beteiligten Redakteuren und unserem Art-Direktor Volker Hilbel stark genug, um ohne zu zögern die Platine mit dem in Neon-Röhren symbolisierten neuen Bus zu durchbohren. Daß die AT-Welle gestorben ist, halten alle Auguren für unwahrscheinlich. IBM will die vertikale Verbindung: seine neuen PCs in die Welt seiner Großrechner integrieren. Und MS-OS/2 läuft ab 80 286 aufwärts, also auf jedem AT. Auch von mc wird man in Bezug auf die klassische Linie der PCs noch viel hören. Aber, Hauptthema in diesem Heft: Der neue Bus von IBM und sein logisches und zeitliches Verhalten.

ie Welt besteht nicht nur aus IBM allein. Apple faßt in der Szene in der Bundesrepublik wieder Fuß. Apple, in den USA vor allem auf dem Gebiet „Desktop Publishing" erfolgreich, hat den Macintosh II mit dem Nubus ausgerüstet. Korrekterweise müßte man Apple-Nubus dazu sagen, denn erfunden haben diesen Bus die Techniker von Texas Instruments und Apple hat ihn für seine Zwecke abgewandelt. Wir beschreiben beide Bus-Systeme in diesem Heft, weil sie in typischen Vertretern der neuen Mikrocomputer-Generation implementiert sind. Sie machen aus diesen Computern komplexe Systeme, deren Komponenten sich dynamisch eingliedern lassen. Die neue Generation schließt eng zu den Minis auf, was die universelle Erweiterbarkeit und modulare Konfigurierbarkeit angeht. Und die Größe.

Zu unserem Biberwettbewerb wollte ich hier schon einen Zwischenstand angeben. Wir haben aber so viele Biber eingesandt bekommen, daß die Auswertung noch etwas Zeit benötigt.

Ich war auf einem sehr interessanten Pressegespräch, das von der Bundespost und Siemens gemeinsam abgehalten wurde. Einigen ausgewählten Journalisten hat die Bundespost seit vergangenem Oktober ein Bildschirmtext-Endgerät modernster Prägung hingestellt, mir zum Beispiel ein Bitel von Siemens. Jetzt wollte man, zusammen mit Anbietern hören, wie Journalisten Btx nutzen. Ich habe im Gespräch engagiert die Meinung vertreten, daß Bildschirmtext von heute aus betrachtet ein Medium sei, das vielleicht einen richtigen Gedanken so gnadenlos in Cepttechnik realisiere, daß Menschen wie ich Schwierigkeiten mit der Performance und dem inhaltlichen Sinn der Sache bekommen. Als ich dann noch sagte, daß die Veranstaltung doch wohl zeige, daß hier ein Medium seinen Inhalt suche (1 Mrd. DM öffentliche Investitionen!), waren alle sehr betrübt. So grundsätzlich wollte das keiner der Beteiligten hören, es war offenbar zu vorlaut. Lieber wollte man über die kleinen Mängel reden, daß vielleicht die Tastatur der Geräte zu klein ist, weshalb man denn nicht einfach einen normalen Matrixdrukker an ein Bitel hängen könne – und über die Vorteile, die durchaus bestehen, denn man kann wirklich gute Informationsdienste in Btx aufziehen – das sei zugegeben. Vielleicht bekomme ich kurze Vorschläge, was der eine oder andere in Btx für einen Sinn sieht, das würde sich in der Leserbriefabteilung gut machen!

Ihr

Ulrich Rohde

mc-inhalt

mc-editorial

Liebe Leser 3

mc-briefe 6

mc-info 10

Spruch des Monats 35

Zitat des Monats 41

Desktop Publishing für Einsteiger 58

Hochintegrierte PC-Bausteine 96

Der Markt für ISDN-Produkte 108

mc-bücher 26

Neues vom Buchmarkt 105

mc-grundlagen

Apple NuBus 32
Mit dem Macintosh II erfüllt Apple endlich den Wunsch vieler Macintosh-Anwender nach einer offenen System-Architektur. Der 68020-Computer ist mit sechs Steckplätzen nach dem von Texas Instruments entwickelten NuBus-Standard, den wir ausführlich beschreiben, erweiterbar

„Monte Carlo" im Computer 36
Bei Computern ist im allgemeinen nichts dem Zufall überlassen. Werden zur Simulation von physikalischen Vorgängen mit einem Computermodell zufällig verteilte Werte benötigt, gibt es verschiedene Verfahren der Erzeugung

Das ist der IBM-Mikro-Kanal-Bus 42
Im zweiten Teil unserer sehr detaillierten Schilderung des Systembusses der neuen Computerfamilie PS/2 von IBM geht es um die automatische Systemkonfiguration, die Arbitration und das Zeitverhalten

Einführung in Unix 50
Ein Unix-System wird über die kommandoorientierte Benutzeroberfläche Shell bedient. In unserer Unix-Serie wird die Bourne-Shell erklärt

Assembler-Inlines in C unter CP/M-68k 53
Auch der C-Compiler von Digital Research für das Betriebssystem CP/M-68k erlaubt das Einbinden von Assemblerbefehlen

Transputer 54
Daß Transputer-Systeme bedeutend schneller als „normale" Computer sind, weiß jeder. Wie sie funktionieren und daß sie neue Denkansätze bei der Programmierung erfordern, steht in diesem fundierten Beitrag

mc-soft

mc-CP/M-Computer mit Uhr 59
Ein nicht benutzter Timer des Z80-STI-Bausteins auf der Term1-Platine wird durch geschickte Programmierung zur Uhr

SCEDIT 62
Ein Bildschirm-Editor für den mc-68000-Computer unter CP/M-68k, der auf dem Monitorprogramm in der Version 1.4 und einer IBM-Tastatur aufbaut

Vier Bildschirmseiten in Turbo-Pascal 69
Mit der hier beschriebenen Änderung im Turbo-Pascal können Programme für die CGA-Karte vier Bildschirmseiten unabhängig voneinander nutzen

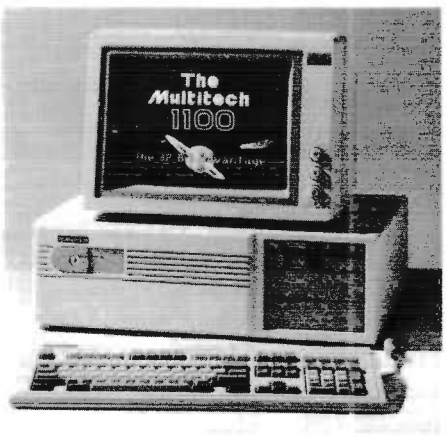
Titelfoto: Klaus Hager

Apple NuBus und IBM-Mikro-Kanal

Aufbauend auf einem von Texas Instrument definierten synchronen Standard-Bus hat Apple den Apple NuBus in den Macintosh II integriert; IBM setzt auf eine Eigenentwicklung. Wir durchleuchten beide Buskonzepte ab **Seite 32**

Tests ...

Der Sanyo MBC-16 PC, das echtzeitfähige Betriebssystem QNX und Turbo-C zeigen im mc-Test, was sie können. Der schnellste aller ATs ist zur Zeit der Multitech-1100. Unsere Meinung zu diesem Computer finden Sie ab **Seite 98**



IBM PS/2 Modell 60

Der stärkste lieferbare Vertreter aus der Familie der neuen Computer von IBM ist für die derzeit vorhandene Software zu leistungsfähig. Wer trotzdem schon jetzt in den sich zur Abfahrt bereitmachenden Zug einsteigen möchte, findet einige Tips ab **Seite 106**

Screen Saver für Macintosh

Neben der detaillierten Beschreibung des Apple NuBus, der im Macintosh II den Einsatz von sechs Steckkarten erlaubt, beschreiben wir, wie der Bildschirm des Macintosh vor dem Einbrennen geschützt wird. Das Programm mit ausführlicher Beschreibung finden Sie ab **Seite 76**

MS-DOS mit Komfort 74
Wer seinem Computer in einer Kommandozeile mehrere Befehle geben möchte, sollte diesen Beitrag lesen

Assemblerprogramm für den Macintosh 76
Ein nützliches Programm für den Macintosh, das den Bildschirminhalt nach dem Ablauf einer einstellbaren Zeit sichert und den Bildschirm löscht, wird mit vielen Interna des Macintosh vorgestellt

GW-Basic ruft MS-DOS 80
Wir stellen eine Assembler-Routine vor, mit der MS-DOS-Funktionen von Basic aus aufgerufen werden

Arcus-Sinus in C 81
Ein Modul für den C-Compiler des CP/M-68k

Grenzgänger 82
Bei einem AT kann man mit wenig Aufwand die 1-MByte-Grenze überschreiten. Wir stellen ein Beispiel in Turbo-Pascal vor-

Einfache Schwarzweiß-Grafik unter C 86
Eine Graphic Toolbox mit fünf Funktionen für den IBM PC und dazu kompatible Computer, die mit wenig Programmieraufwand ansehnliche Ergebnisse bringt

mc-Allprog 88
Der dritte Teil der Beschreibung unseres universellen Programmiergerätes geht auf die Steuersoftware unter CP/M und PC-DOS ein

mc-hard

Universal-Interface 93
Mit einer Steckkarte lassen sich 6502- und Z80-Bausteine gleichermaßen in unterschiedlichen Computern einsetzen. Ein nützlicher Artikel für Apple-II-, C-64- und MZ-700-Fans

mc-test

Multitechs 386er 98
Der Multitech-1100 ist 18mal schneller als ein PC und kommt mit viel Software und Dokumentation

C mit Umgebung 100
Turbo-C ist da und alle Programmierer sind sich bei der Beurteilung einig: Einfach Spitze!

Exklusiv für Profis 102
Bereits auf einem PC mit 512 KByte RAM kann man ein echtzeitfähiges Multitasking-Betriebssystem einsetzen, das viele Ähnlichkeiten zu Unix aufweist

Klein – aber oho 104
Der Sanyo MBC-16 PC wird wahlweise mit 4,77 MHz oder 8 MHz getaktet und mit einer vorbildlichen Dokumentation geliefert

Aufgebohrter AT 106
Erste Eindrücke von IBMs Personal System /2 Modell 60 mit Programmen zum Übertragen von Text- und Programm-Dateien

mc-markt 110

mc-vorschau 142

Impressum 140

```
dim stab(0:33); stab(i)=sin(Π/64*i), i=0,33;
anfang: z=x/(Π/64); i=int(z); dx=z-i;
f1=stab(i+1)-stab(i); f2=(stab(i+2)-stab(i))/2;
a=f2-f1; b=f1-a; s=stab(i)+(a*dx+b)*dx; fertig.
```

Berechnung von s = sin (x) für 0 ≤ x ≤ pi/2

Sin und cosin

Mit großem Interesse habe ich in mc 6/87 Anton Obermaiers Artikel „Sin und cosin mit Integerarithmetik" gelesen. Die von Herrn Obermaier gewählten Taylor-Entwicklungen zur Darstellung der beiden trigonometrischen Funktionen haben den Nachteil, daß sie zwar in der Nähe der Entwicklungsstelle x=0 sehr gute Näherungswerte liefern, daß aber in der Nähe von x = pi/2 eben noch Terme neunter Ordnung in x berücksichtigt werden müssen, um Genauigkeiten von etwa 10^{-4} zu erreichen. Günstiger fährt man mit Polynomdarstellungen, wie sie zum Beispiel in Abramowitz/Stegun, Handbook of Mathematical Funktions, Formeln 4.3.96 und 4.3.98 angegeben sind, oder mit quadratischer Interpolation in einer Sinustabelle, die für die angestrebte Genauigkeit im Winkelbereich von Ø bis pi/2 nur in 32 Teile unterteilt werden muß. Letztere Möglichkeit sei durch den Pseudocode zur Berechnung von s=sin(x) für 0≤x≤pi/2 im Bild illustriert. Damit ist die Sinusberechnung im wesentlichen mit zwei Multiplikationen abgehandelt. Die Division bei der Berechnung von z kann durch Shifts erledigt werden, wenn man davon ausgeht, daß x ohnehin in einer auf Vielfache oder Bruchteile von pi normierten Darstellung vorliegt. Weitere Ersparnisse an Rechenzeit ergeben sich, wenn man die Koeffizienten a und b zu den Tabelleneinträgen auch noch vorab berechnet und mit abspeichert.

Dipl.-Phys. Johannes E. Philipp, 5067 Kürten 3

Hydra kaum zu töten

Mit einem mir in meiner Firma zur Verfügung stehenden HP9020-Rechner habe ich nachts mit einem Programm, das im Wesentlichen dem in Bild 6 aus dem Beitrag „GOTO und Standard-Basic" in mc, Ausgabe 6, 1987 entspricht, nach weiteren Zahlen gesucht. das Ergebnis finden Sie im Bild unten rechts. Ausgedruckt ist jeweils nacheinander in einer Zeile die gefundene Zahl Y, deren Dualdarstellung (ab 32000 wohl nicht ganz korrekt), die Anzahl der Durchläufe X, die größte bis einschließlich Y aufgetretene Zahl A (max. Anzahl der Hydra-Köpfe) und die Zeit ab Programm-Start. Untersucht wurden alle ungeraden Zahlen von 9 bis 5649499. Der Rechner hat einen 32-Bit-Prozessor, keinen Arithmetik-Coprozessor, arbeitet mit einer Kompilation des sogenannten HP-Rocky-Mountain-Basics und benutzt eine 64-Bit-Real-Darstellung, die Integer-Zahlen bis 2 hoch 35 zu unterscheiden gestattet. Der kritische Punkt beim Überlauf ist nicht die Testzahl Y selbst, sondern die Hydra-Kopf-Zahl A, die weit größer werden kann als Y. Einen Überlauf fängt man doch wohl am elegantesten universell mit einer Formulierung ab wie zum Beispiel IF A = A+1 THEN STOP.

Dr. Gunter Born, 8011 Vaterstetten

Turbo-Pascal und Hercules-Karte

Ich bin ein regelmäßiger Leser Ihrer Zeitschrift und finde sie sehr gut. Ich würde aber auch gern aktiv etwas beitragen (ich erstelle kleinere Programme in Turbo-Pascal und 8088-Assembler). Es würde mich sehr freuen, wenn Sie mir Ihre Autoren-Richtlinien zuschicken würden. Außerdem möchte ich noch eine Verbesserung für den Artikel „Turbo-Pascal bedient Hercules-Karte " aus mc 2/1987 vorschlagen. Wenn man bei der Berechnung des ByteOffset die Zeile

ByteOffset := $2000·(Y MOD 4) +90·(Y DIV 4)+(X DIV 8);

durch

ByteOffset := $2000·(Y MOD 4)+90·(Y SHR 2)+(X SHR 3);

ersetzt, läßt sich die (leider sehr langsame) Ausgabegeschwindigkeit eines Punktes um ca. 15 % steigern. Das kommt daher, daß die Verschiebe-Befehle wesentlich schneller ausgeführt werden als die DIV-Befehle.

Stefan Buchholz, 7030 Böblingen

9	00000000000000000000000000001001	19	52	00:00:00
19	00000000000000000000000000010011	20	160	00:00:00
25	00000000000000000000000000011001	23	160	00:00:00
27	00000000000000000000000000011011	111	9 232	00:00:00
55	00000000000000000000000000110111	112	9 232	00:00:00
73	00000000000000000000000001001001	115	9 232	00:00:01
97	00000000000000000000000001100001	118	9 232	00:00:01
129	00000000000000000000000010000001	121	9 232	00:00:01
171	00000000000000000000000010101011	124	9 232	00:00:01
231	00000000000000000000000011100111	127	9 232	00:00:02
313	00000000000000000000000100111001	150	13 120	00:00:02
327	00000000000000000000000101000111	143	13 120	00:00:02
649	00000000000000000000001010001001	144	41 524	00:00:04
703	00000000000000000000001010111111	170	250 504	00:00:04
871	00000000000000000000001101100111	178	250 504	00:00:05
1 161	00000000000000000000010010001001	181	250 504	00:00:06
2 223	00000000000000000000100010101111	182	1 276 936	00:00:12
2 463	00000000000000000000100110011111	208	1 276 936	00:00:14
2 919	00000000000000000000101101100111	216	1 276 936	00:00:16
3 711	00000000000000000000111001111111	237	1 276 936	00:00:21
6 171	00000000000000000001100000011011	261	8 153 620	00:00:36
10 971	00000000000000000001010101011011	267	27 114 424	00:01:08
13 255	00000000000000000001100111100111	275	27 114 424	00:01:23
17 647	00000000000000000010001001101111	278	27 114 424	00:01:53
23 529	00000000000000000010110111101001	281	50 143 264	00:02:34
26 623	00000000000000000011001111111111	307	106 358 020	00:02:56
34 239	00000000000000001000001011011111	310	121 012 864	00:03:52
35 655	00000000000000001000010111010001	323	121 012 864	00:04:02
52 527	00000000000000001010011010010111	339	121 012 864	00:06:09
77 031	00000000000000010001011001110011	350	593 279 152	00:09:21
106 239	00000000000000011000111101111111	353	1 570 824 736	00:13:17
142 587	00000000000000100001011001111011	374	2 798 323 360	00:18:18
156 159	00000000000000100011000011111111	382	2 798 323 360	00:20:12
216 367	00000000000000110010011010010111	385	17 202 377 752	00:28:47
230 631	00000000000000111000010011100111	442	17 202 377 752	00:30:51
410 011	00000000000001100010000011001101	448	24 648 077 896	00:57:31
511 935	00000000000001111010011111011111	469	24 648 077 896	01:13:06
626 331	00000000000010011000011101001011	508	24 648 077 896	01:30:53
837 799	00000000000011001010010010100111	524	56 991 483 520	02:04:18
1 117 065	00000000001000100000101110001001	527	90 239 155 648	02:49:21
1 501 353	00000000001011010110100010101001	530	151 629 574 372	03:52:33
1 723 519	00000000001101000100110001111111	556	151 629 574 372	04:29:33
2 298 025	00000000010001100001000010101001	559	156 914 378 224	06:06:50
3 064 033	00000000010111010100000011100001	562	622 717 901 620	08:19:08
3 542 887	00000000011011000000111101100111	583	622 717 901 620	09:43:07
3 732 423	00000000011100010111001111000111	596	622 717 901 620	10:16:31
5 649 499	00000000101011000011010001011011	612	1 318 802 294 932	16:05:11

Hydra kaum zu töten ...

```
lerne unter :varia :varia1 :lauf
 wenn :varia1\2=1 [setze "varia1 :varia1*3+1][setze "varia1 :varia1/2]
 setze "lauf=lauf+1
 wenn :varia1=1 [pruefe :lauf>200  {*sucht natuerliche Zahlen, die mehr als}
                                   {* 200 Schleifendurchlaeufe erzeugen.}
                        ww [dz :varia rk]
                        wf [rk]
unter :varia :varia1 :lauf
ende
vi
setze "varia 0
wh 10000 [setze "varia :varia + 1 setze "varia1 :varia unter :varia :varia1 0]
{*Das Programm untersucht den Zahlenraum bis 10000}
```

Hydra 2

In der Beilage erhalten Sie einen Computer-Output zum Hydra-Problem. Das Programm entspricht im wesentlichen Bild 6 des genannten Artikels und berechnet für die aufgeführten Zahlen z die maximale Anzahl Durchläufe d und die dafür benötigte Zeit. Verwendeter Computer: Atari 1040 ST. Programmiersprachen: C für Ausgabe und Timing ab Start, 68000-Assembler in 64-Bit Ganzzahlarithmetik für die innerste Schleife.

Bemerkung: 32 Bit genügen nicht, da bereits in diesem Wertebereich Zwischenwerte 1 000 000 000 auftreten!

Waldemar Wasserbauch, Ch Zürich

Hydra 3

Ich habe als Besitzer eines NDR-Klein-Computers mit CPU 68008 ein wenig bei dem Hydra-Problem aus mc 6/1987 mitprogrammiert. Anbei übersende ich Ihnen ein Gosi-Programm, das das Programm aus Bild 5 im Artikel GOTO und Standard-Basic nachempfindet. Die Laufzeiten: Erster Wert 2 min 41 s, letzter Wert 13 min 15 s.

Bernd Ludwig, 2330 Eckernförde

```
Ausdruck von    C:\HYDRA1.C   am 10.6.1987

          /* HYDRA Test Abbrechen einer Folge */
#include <lang.h>
#define TIMEO   long t1,t2,*ptr; gettime() {*ptr=*(long *)0x462;}
#define TIME1   ptr = &t1; Supexec(gettime)
#define TIME2   ptr = &t2; Supexec(gettime)
#define TIME    printf(" %13.2f sec",(float)(t2-t1)/70.0)

TIMEO

main(argc,argv)
{  int d, dmax;              /*Schleifenzähler und Maximalwert      */
   long zin;                 /*Startwert                            */
   register long zb,za;      /*Zwischenwert      , 64 bits !        */
   setbuf(stdout,NULL);
   zin = 1L;                 /*Start mit 1                          */
   dmax = 1;
   TIME1;
   FOREVER                   /*nicht abbrechende Schleife           */
   {  za = 0L;
      zb = zin;
      d = 0;
      asm
      {  LA3:    ADDQ.W   #1,-2(A6)   ; inkrementiere d
                 BTST     #0,D7       ; z ungerade ?
                 BEQ      LA2         ;   ja
                 MOVE.L   D6,D0       ;   nein, multipliziere mit 3
                 MOVE.L   D7,D1
                 ADD.L    D7,D1       ; 1. Addition
                 ADDX.L   D6,D0
                 ADD.L    D1,D7       ; 2. Addition
                 ADDX.L   D0,D6
                 ADDQ.L   #1,D7       ; addiere 1 zum Resultat
                 BCC      LA3         ; neuer Anlauf
                 ADDQ.L   #1,D6       ; Carry Bit
                 BRA.S    LA3         ; neuer Anlauf
         LA2:    LSR.L    #1,D6       ; Division durch 2 bei z gerade
                 ROXR.L   #1,D7       ; durch Rechtsschieben
                 CMPI.L   #0,D6       ; ist z = 1 ?
                 BNE.S    LA3         ;   nein, neuer Anlauf
                 CMPI.L   #1,D7
                 BNE.S    LA3         ;   nein, neuer Anlauf
      }
      if (d > dmax)
      {  printf("\n\r z = %81d",zin); /*Ausgabe                      */
         printf("  d = %8d", d);
         TIME2;
         TIME;
         dmax = d;
      }
      zin++;                         /*nächster Wert                */
   }                                 /*Schleifenende                */
}
```

```
      2463      7515
      2919      7785
      3695      7963
      3711      8315
      4233      8351
      4255      8425
      4379      8455
      4926      8466
      4927      8467
      5543      8510
      5567      8511
      5673      8697
      5799      8699
      5838      8758
      5839      8759
      6171      8959
      6569      9225
      6591      9257
      6919      9852
      6943      9853
      7390      9854
      7391      9887
      7422      9897
      7423
```

```
z =          1  d =          3          .01 sec
z =          3  d =          7          .06 sec
z =          6  d =          8          .09 sec
z =          7  d =         16          .13 sec
z =          9  d =         19          .16 sec
z =         18  d =         20          .20 sec
z =         25  d =         23          .46 sec
z =         27  d =        111          .86 sec
z =         54  d =        112         1.26 sec
z =         73  d =        115         1.66 sec
z =         97  d =        118         2.06 sec
z =        129  d =        121         2,46 sec
z =        171  d =        124         2.86 sec
z =        231  d =        127         3.26 sec
z =        313  d =        130         3.66 sec
z =        327  d =        143         4.06 sec
z =        649  d =        144         4.51 sec
z =        703  d =        170         4.86 sec
z =        871  d =        178         5.26 sec
z =       1161  d =        181         5.71 sec
z =       2223  d =        182         6.97 sec
z =       2463  d =        208         7.27 sec
z =       2919  d =        216         7.83 sec
z =       3711  d =        237         8.80 sec
z =       6171  d =        261        11.81 sec
z =      10971  d =        267        18.04 sec
z =      13255  d =        275        21.16 sec
z =      17647  d =        278        27.17 sec
z =      23529  d =        281        35.44 sec
z =      26623  d =        307        39.90 sec
z =      34239  d =        310        50.94 sec
z =      35655  d =        323        53.11 sec
z =      52527  d =        339        78.44 sec
z =      77031  d =        350       116.59 sec
z =     106239  d =        353       163.56 sec
z =     142587  d =        374       223.69 sec
z =     156159  d =        382       246.49 sec
z =     216367  d =        385       349.27 sec
z =     230631  d =        442       373.99 sec
z =     410011  d =        448       694.11 sec
z =     511935  d =        469       881.69 sec
z =     626331  d =        508      1095.40 sec
z =     837799  d =        524      1498.03 sec
z =    1117065  d =        527      2040.93 sec
z =    1501353  d =        530      2803.20 sec
z =    1723519  d =        556      3249.46 sec
z =    2298025  d =        559      4423.79 sec
z =    3064033  d =        562      6021.09 sec
z =    3542887  d =        583      7035.96 sec
z =    3732423  d =        596      7439.06 sec
z =    5649499  d =        612     11588.73 sec
z =    6649279  d =        664     13790.92 sec
z =    8400511  d =        685     17700.47 sec
z =   11200681  d =        688     24054.24 sec
```

Die NCC '87 in Chicago

„Entdecken Sie die Macht der Information" lautete das Motto der diesjährigen National Computer Conference (NCC) in Chicago (*Bild 1*). Ungefähr 170 Aussteller stellten im McCormick Place, dem größten Kongreßzentrum der Welt, ihre Neuheiten aus. Parallel zur Ausstellung wurden zahlreiche technische Seminare abgehalten. In diesen wurden interessante Themen behandelt, wie z. B. die Weiterentwicklung von Prolog, heutige und zukünftige Mikrocomputer-Architekturen, Netzwerk-Techniken und Architekturen der Supercomputer.
Leider befindet sich die NCC im einem Abwärtstrend. Nur acht Aussteller stellten Mikrocomputer vor. Darunter die Firmen IBM, NCR und Fujitsu. Die Zahl der Minicomputer- oder gar Großcomputer-Hersteller war noch geringer. Firmen wie DEC, Apollo, Sun und die PC-Hersteller Commodore und Atari fehlten. Dagegen waren zahlreiche neue Firmen vertreten.

Miniatur-Laufwerk von Alps

Ein 3½-Zoll-Laufwerk mit 2 MByte Speicherkapazität stellte Alps OEM vor (*Bild 2*). Der Clou: das Laufwerk ist nur 2,54 cm hoch! Dieses Laufwerk gehört zur DFP-Serie, die sowohl in TTL- als auch in CMOS-Technik geliefert werden kann. Eine portable 20-MByte-Festplatte von Alps vereinigt die Kapazität einer Festplatte mit der Austauschbarkeit einer Diskette. Das Laufwerk paßt in den 5¼-Zoll-Gehäuseausschnitt eines PCs.
Auf dem IBM-Stand war der 386-PC der PS/2-Serie, das Modell 80, zu bewundern. IBM hat eine verbesserte Version des NetView/PC-Programms angekündigt. NetView/PC ermöglicht in Zukunft die Kommunikation mit Computern, die nicht der IBM-SNA-Welt (SNA: Systems Network Architecture) angehören. Dazu standardisiert IBM das Format der Netzwerk-Botschaften, so daß diese von der Netzwerk-Software anderer Hersteller verstanden werden.

Handwriter ersetzt Tastatur und Maus

Eine neues und für manche Anwendungen vielleicht sogar revolutionäres Eingabemedium für PCs ist der Handwriter. Mit diesem Gerät kann man tatsächlich handschriftliche Eingaben in den PC vornehmen. Im Prinzip ähnelt es dem grafischen Tablett, ein Eingabemedium, das professionellen Grafik-Anwendern schon lange bekannt ist. Der Handwriter besteht aus einem Schreibstift, einem Tablett und einer Papierschablone auf der z. B. ein Feld als „Eingabetaste" markiert ist. Entwickelt wurde der Handwriter von dem renomierten Stanford Forschungsinstitut, das auch erheblich zur Entwicklung der Maus beitrug. Voraussichtlich bringt Computer Intelligence Corporation den Handwriter im Spätsommer auf den Markt.
Mit dem Handwriter ist es möglich, einen Computer ohne Tastatur und Maus zu betreiben. Interessante Anwendungen ergeben sich im Bereich Desktop Publishing. Illustrationen und Cartoons können damit direkt in das System eingegeben werden. Für japanische und chinesische

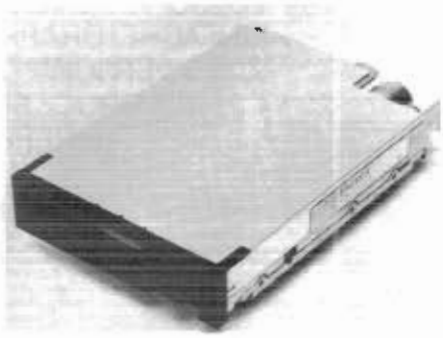


Bild 2. Viel Speicher auf wenig Platz: DFP 723D von Alps OEM

Schriftzeichen dürfte der Handwriter das einzige akzeptierbare Eingabemedium sein. Gut geeignet ist das System zur Unterschriftenerkennung im Bankenbereich und in der Zugangskontrolle. Im Schulbereich ließe es sich sogar zur Verbesserung der Handschrift einsetzen, denn schlechte Schrift wird vom System nicht erkannt.
Der von der IEEE-Gesellschaft veranstaltete Micro Mouse Wettbewerb zog die Besucher wie ein Magnet an. Ein Miniaturroboter (Micro Mouse) mußte sich durch ein Labyrinth bewegen. Hier durften die Methoden der künstlichen Intelligenz im kleinen angewendet werden. Die meisten Probleme hatten die Miniaturroboter allerdings mit der Mechanik!
Um die zukünftigen NCCs für Aussteller und Besucher wieder attraktiver zu machen, wird ab nächstes Jahr die National Computer Exposition (NCE) in die NCC integriert werden. Mit der Organisation der nächsten NCC, die 1988 in Los Angeles stattfinden wird, ist ISA Services beauftragt worden. Insbesondere der in der Vergangenheit stark vernachlässigte PC-Bereich soll auf der nächsten NCC verstärkt angesprochen werden.

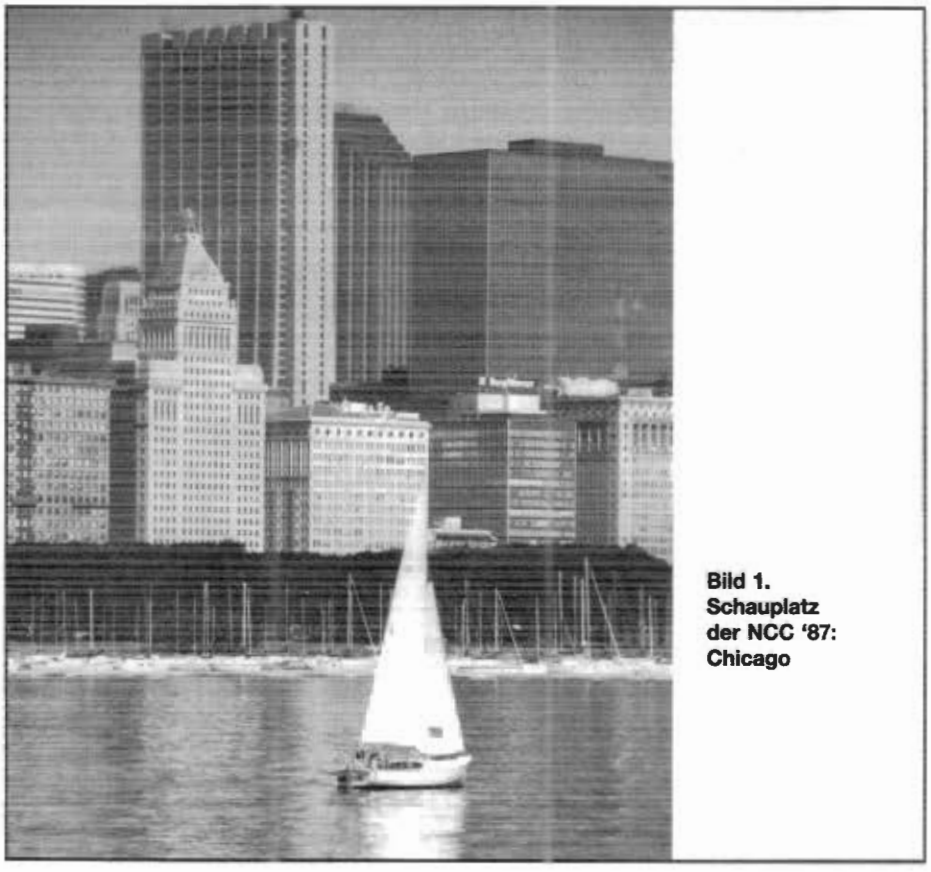


Bild 1. Schauplatz der NCC '87: Chicago

Seminare

Mit den Themen Software-Engineering, -Projektmanagement und -Qualitätssicherung beschäftigen sich Seminare, die von der Ambit GmbH aus 8011 Hohenbrunn Ende Oktober bzw. Anfang November in Zürich-Schlieren veranstaltet werden.

Vom 7. bis zum 11. September 1987 findet der Workshop „OCCAM II und die Transputerfamilie" in Karlsruhe statt. Neben einer allgemeinen Einführung wird intensiv an Transputer-Systemen trainiert. Für Fortgeschrittene bietet sich der Workshop vom 26. bis zum 28. Oktober 1987 an. Informieren und anmelden kann man sich bei Gisela Ziegler unter 0721/886291.

Im Produktionstechnischen Zentrum der Technischen Uni Berlin wird ab Mitte Oktober die Seminarreihe „Fortgeschrittene CAD-Technik" gestartet. Über die Veranstaltung, die vom VDI-Bildungswerk Düsseldorf organisiert wird, kann man sich unter 0211/6214201 informieren.

Am 14. September 1987 beginnt das Technologie-Management-Programm der GNB (Gesellschaft für neue Berufe). Hoch- und Fachhochschüler können das Ausbildungsziel „Systemarchitekt" in den Richtungen Telematik, Marketing/Vertrieb oder Knowledge Engineering anstreben. Interessenten wenden sich unter 030/460040 an die GNB.

Ein Grundseminar über den Novix NC 4000 Forth-Prozessor hält die RSO Gesellschaft für technische Kybernetik (089/429188) vom 12. bis zum 14. Oktober 1987 in München ab.

Ein Kurs über digitale Schnittstellen in Computersystemen und Übertragungsnetzen findet am 14. und 15. Oktober 1987 im Haus der Technik in Essen statt. An gleicher Stelle wird am 21. und 22. Oktober 1987 über die Mikroprozessoren MC 6809, MC 68000 und MC 68020 informiert. Das ausführliche Veranstaltungsprogramm gibt es unter 0201/18031.

National Semiconductor bietet Seminare über Mikrocomputersysteme, ASIC-Produkte und andere Fachgebiete im Trainingscenter in Fürstenfeldbruck bei München an. Alle Seminarthemen sind in einer Broschüre zusammengefaßt, die auf Anfrage unter 08141/103477 zugesandt wird.

Termine

Der Computer Club e.V. aus 3057 Neustadt veranstaltet am 24./25. Oktober 1987 die 5. Neustädter Computertage. Anbieter und Entwickler aus dem Großraum Hannover werden ihre Exponate für den Heim- und Hobby-Anwender vorstellen.

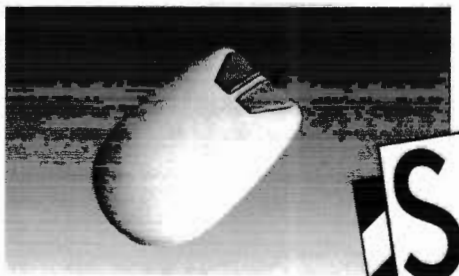
Bei der Technischen Akademie Esslingen in 7302 Ostfildern findet vom 8. bis zum 10. September 1987 das 2. Kolloqium „Software-Entwicklung-Systeme und -Werkzeuge" statt. Informieren und anmelden kann man sich unter Tel. 07 11/3 40 08 23.

Das 13. Euromicro-Weltsymposium wird vom 14. bis zum 17. September 1987 in Portsmouth, Südengland, stattfinden. Einzelheiten über das Programm, in dem unter anderem RISC- und Transputer-Techniken behandelt werden, gibt es bei Dr. H. Schumny, PTB, Bundesallee 100 in 3300 Braunschweig.

Die Ausstellung „Chip '88", die vom 21. bis zum 24. Januar 1988 auf dem Hamburger Messegelände zu sehen ist, wendet sich an Hobby-Elektroniker und Mikrocomputer-Anwender. Auf etwa 6000 Quadratmetern soll ein neuartiger Info-Markt für Elektronik und Software mit einem umfassenden Angebot gezeigt werden.

Zum dritten Mal wird dem Thema „Portables" und „mobile Datenerfassung" eine eigene Veranstaltung gewidmet. Am 15. und 16. September 1987 wird im Kongreß-Centrum der Düsseldorfer Messegesellschaft zu den Entwicklungen auf dem Gebiet der mobilen Computer Stellung genommen. Hersteller, unabhängige Experten und Anwender diskutieren über die weitere technische Entwicklung und gehen auf Fragen bei den Anwendern ein.

Der zweite Euro Software-Kongreß findet vom 7. bis zum 9. Dezember in Karlsruhe statt. Neben Anwendungs- und Systemsoftware werden Produkte für den Bereich Künstliche Intelligenz sowie neue Sprachen der vierten Generation präsentiert. Der Kongreß steht in diesem Jahr unter dem Schwerpunkt Software und Recht. Weitere Informationen gibt es bei der Network GmbH unter Tel. 0 50 33/70 57.



SAYONARA!

RS 232 Mouse
Microsoft-kompatibel **DM 139,–**

Floppy Disks
D 503 V, 360 KB **DM 226,–**
D 509 V, 1,2 MB/360 KB **DM 291,–**

Keyboards
101 **DM 169,–**
84 **DM 113,–**

XT- und AT-Peripherie von MITSUMI. Superqualität zum Superpreis!

Preise incl. MwSt.
– Händleranfragen erwünscht!

MECHATRONIC SYSTEME GmbH
Industriestraße 12
D-7900 Ulm-Lehr
Tel. (0731) 61364

EGA-PC von Schneider

Ab sofort soll der neue PC 1640 von Schneider verfügbar sein, der jetzt in München vorgestellt wurde. In der Grundversion verfügt der PC 1640 über 640 KByte RAM, eine mit 8 MHz getaktete 8086-CPU und einen integrierten Multifunktions-Grafik-Adapter (MIGA). Über Dip-Schalter, die leicht zugänglich an der Gehäuserückseite angebracht sind, wird die Betriebsart Hercules, CGA oder EGA ausgewählt. Neben dem MIGA sind die serielle, die parallele und die Maus-Schnittstelle auf der Hauptplatine integriert. Die drei langen Erweiterungssteckplätze bleiben auch in einem Modell mit Festplatte frei, da für den Laufwerkscontroller ein separater Steckplatz vorgesehen ist. Die Lautstärke des eingebauten Lautsprechers ist einstellbar.

Den EGA-PC 1640 wird es in neun Versionen geben. Das preiswerteste Modell mit einem 5¼-Zoll-Floppy-Laufwerk und einem monochromen Monitor für den Betrieb in der Betriebsart Hercules (720 × 346 Bildpunkte) wird 1698 DM kosten. Mit einer 20-MByte-Festplatte und einem 14-Zoll-EGA-Monitor kostet der PC 1640 4498 DM. Neben dem Hercules- und dem EGA-Monitor wird von Schneider noch ein CGA-Monitor angeboten. Wie schon beim PC 1512, der in den letzten acht Monaten fast 100 000 Interessenten fand und weiter gebaut wird, versorgen die Monitore den PC mit Spannung. Der EGA-Monitor ist mit 256 KByte RAM und einem Lüfter versehen. An Software legt Schneider den neuen Computern MS-DOS 3.2, GEM, GEM Paint und Locomotive Basic 2 bei. Von DOS+, das mit den PC 1512-Modellen ausgeliefert wurde, hat sich Schneider getrennt.

Für Ende 1987 wurde von Schneider ein EGA-AT in Aussicht gestellt, der möglicherweise mit 3½-Zoll-Laufwerken ausgestattet sein wird.

Neben den Personal Computern betreut Schneider auch die Heimcomputerlinie weiter. Für den CPC 464 wird es für etwa 400 DM ein BTX-Terminal geben, das von Schneider in Zusammenarbeit mit Siemens und der Deutschen Bundespost entwickelt wurde. Für das Weihnachtsgeschäft wurden nicht näher beschriebene Aktivitäten im Heimcomputerbereich angekündigt.

Nach Meinung von Schneider ist der professionelle PC-Markt in Deutschland erst zu etwa 10% gesättigt. Um diesen zukunftsträchtigen Markt zu erschließen wurde ein modulares Softwarepaket für den kaufmännischen Bereich angekündigt, das mit Schulung und Hotline-Service günstig angeboten werden soll.

Dem Kommunikationsmedium Fernkopierer möchte Schneider mit „Personal Fax" zum Durchbruch verhelfen. Das System soll eine DIN A4-Seite in 25 Sekunden übermitteln und unter 3000 DM kosten.

Mit dem EGA-PC 1640 möchte Schneider an den Erfolg des PC 1512 anknüpfen. Der Computer ist in neun Versionen erhältlich

Microsoft und Olivetti arbeiten zusammen

Eine verstärkte Zusammenarbeiit haben die deutsche Olivetti GmbH und die Microsoft GmbH bestätigt. Ab sofort werden alle Produkte, die der PC-Software-Hersteller vertreibt, auch über die deutsche Olivetti erhältlich sein. Die Kooperation schließt auch die Olivetti-Ausbildungszentren ein, die verstärkt Schulungen für Microsoft-Produkte anbieten werden. In Italien wurde von den beiden Unternehmen eine Firma zur Entwicklung von CD-ROM-Applikationen gegründet.

C-TOOLS

Die C-Tools sind nicht nur *direkt einsetzbar* (für **die meisten C-Compiler** z.B. Lattice-C und Microsoft-C), sondern verstehen sich auch als Know how-Tools:

Ausführliche *Begleitdokumentationen* liefern Ihnen detaillierte Informationen und erklären jedes Statement der C-TOOLS.

C-TOOLS Package # 1: Routinen für den Zugriff auf *sämtliche Systemeinheiten* von IBM-Personalcomputern und Kompatiblen, auf die Funktionen des *ROM-BIOS* und des Betriebssystems *DOS* für die Programmiersprache *C* im **deutsch kommentierten** *Source-Code* und im *Objekt-Code*.

Das 1. Package der C-TOOLS enthält über 100 Zugriffsroutinen auf Platte, Bildschirm, Tastatur, Drucker, Lautsprecher, den asynchronen Kommunikationsadaptor und weitere Tools. Soweit möglich, werden die Zugriffsroutinen jeweils auf *allen 3 Zugriffsebenen* zur Verfügung gestellt: auf Programmiersprachen-Ebene in C, auf der Betriebssystem-Ebene v. DOS u. auf BIOS-Ebene. Für alle BIOS-Zugriffe gibt es *assemblersprachliche Schnittstellen*, die auch mit anderen Programmiersprachen verwendet werden können.

Außerdem werden Ihnen unterschiedliche Verfahrenstechniken erklärt, z.B. für *schnelle, störungsfreie Bildschirmausgaben, Bildschirmfenster, Scrolling* etc.; Sie erhalten ein *Synthesizer-Programm*, mit dem Sie auf Ihrer Tastatur beliebige *Tonmuster* oder *Melodien* spielen und diese dann direkt in Ihr Programm einbauen können; Sie erhalten *Druckroutinen* für millimetergenaues Drucken in Vordrucke und für Graphik-Drucken u.v.m.

Preis: 632,70 DM

C-TOOLS Package # 2:

Datenorganisation und Speicherkonzepte, Sortierverfahren, Suchverfahren, Filter für die Programmiersprache C im deutsch kommentierten Source- und Objektcode.

Das Package # 2 enthält in der vorliegenden Version Routinen für:

- interne Datenorganisation/Speicherkonzepte: Listen, Stacks, Hashing inclusive aller Grundoperationen (z.B. Element einfügen, löschen, Position ermitteln etc.).
- Dateiorganisation und -zugriffe: sequentielle, "hashed" und indizierte Dateien
- Arbeitsspeicher- interne, externe und intern/extern-kombinierte Sortierverfahren
- Filter (z.B. variable lexikalische Sortierung, Dupletten-Filter, Dateiverschlüsselung, Spaltenanordnung etc.) **Preis: 855.– DM**

C-TOOLS Package # 3: Ein Generator für dialogorientierte Programmsysteme incl. **Windowing**.

Die überwiegende Anzahl von Anwendersystemen ist heute dialogorientiert. Die Gestaltung der Benutzerschnittstelle ist in erster Linie verantwortlich für die sog. Benutzerfreundlichkeit eines Programmsystems und bestimmt damit maßgeblich dessen Marktchancen. Die Gestaltung d. Benutzerschnittstellen wird deshalb immer trickreicher u. komfortabler. Die Programmierung solcher dialogorientierter Programmsysteme verlangt jedoch dem Programmierer einen großen Aufwand an Zeit u. Arbeit ab. Diese Programmierarbeiten erheblich zu reduzieren, ist die Aufgabe eines Dialogsystem-Generators.

Der Dialogsystem-Generator kann: Graphik- u. Textmodus, Manipulation der Bildschirmattribute, Windows/Pull-Down-Menues, Ein-/Ausgabefelder, Cursormanagement, Dialog- u. Aktionssteuerung.

Preis: 855.– DM

Unterstützte Graphik-Karten: CGA, Hercules, EGA, Olivetti, IBM Professional u.a.

C-Tools Package # 4: Graphik

Das Package enthält die wesentlichen grafischen Grundfunktionen in Quell- u. Objektcode zusammen m. ausführlichen Kommentaren innerhalb u. außerhalb d. Listings. Über 100 Einzelfunktionen in C u. Assembler für

- die Initialisierung der Grafikkarten;
- schnelles Zeichnen von Geraden, Kreisen, Ellipsen, Kreis- und Ellipsenbögen;
- das Ausfüllen von Polygonen (Fill) und konvexen Figuren (Paint) mit unterschiedlichen Füllmustern;
- die Definition von Windows und Viewports;
- die Erzeugung v. Kurven m. Hilfe kubischer B-Splines;
- Textausgaben im Grafikmodus.

z.Z. werden die folgenden Karten unterstützt: Hercules, Olivetti monochrom, CGA, EGA.

Preis: 855,– DM

C-Trainer

Lernen Sie C richtig von Anfang an!

Besonders geeignet für Schulungszwecke und C-Anfänger (C-Interpreter mit Tutorial-Programmierbuch – ein kompletter C-Lernkurs). Der C-Trainer ist eine neue, sehr effektive Methode, um C zu lernen oder sein Wissen zu erweitern. Der C-Trainer besteht aus drei Teilen: Turorial-Buch, C-Interpreter und eine C-Programmbibliothek.

Der C-Interpreter erlaubt eine hervorragende Kontrolle über die Ausführung eines C-Programmes und besitzt große Vorteile bei der Entwicklung von C-Programmen. Das Programm kann an einem beliebigen Punkt gestoppt werden, die Werte aktiver Variablen können eingesehen und geändert werden.

Programmänderungen sind sofort und ohne Compilieren und Linken möglich.

Separater oder gemeinsamer Trace für Funktionsaufrufe, Statements und Expressions ist möglich.

Verfügbar für: IBM/PC 285,– DM, Macintosh 285,– DM, Sun 513,– DM, MicroVAX (Unix o. VMS) 513,– DM, Pyramid 1425,– DM, VAX 11/700 (Unix o. VMS) 969,– DM, C-Tutorial-Buch 58,85 DM

(Alle Preise zzgl. Verpackung und Versand).

Der C-Trainer ist ein Produkt der Catalytix Corp.

Die C-Tools sind Produkte des:

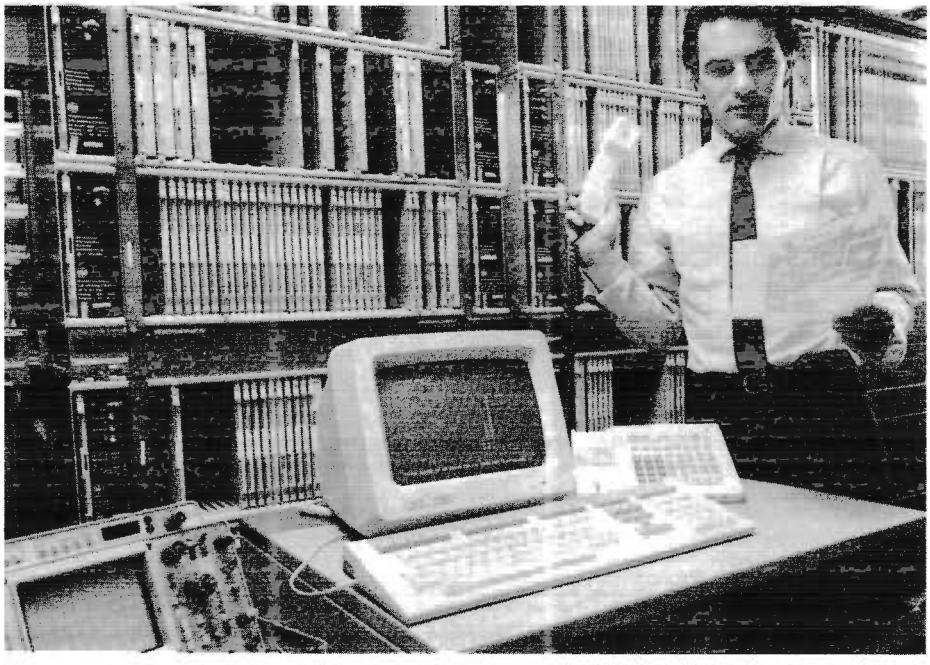
ECO Institut, Postfach 1158, D-8411 Lappersdorf, Telefon (0941) 82509

X-Tools erhalten Menüsteuerung und Fenstertechnik

Mit einer neuen X-Tools-Version für die Sprache C kommt die Nürnberger Ingenieurgesellschaft AiD auf den Markt. Die Version 3.0 unterscheidet sich von der Vorgängerversion 2.2 im wesentlichen durch eine vereinfachte Benutzeroberfläche. Über eine interaktive Menüsteuerung wird der Anwender durch das Tool-Paket geführt. Die Help-Funktion von Version 3.0 kann jederzeit aktiviert werden.
Auch die Fenstertechnik ist neu. Es lassen sich verschiedene Diagramme in Fenstern darstellen, wobei das Fenster, in dem gerade gearbeitet wird, jeweils im Vordergrund erscheint. Auf ein und dasselbe Struktogramm können mehrere Fenster gelegt werden, so daß es sich aus verschiedenen Perspektiven betrachten und bearbeiten läßt. Die neue X-Tools-Version steht zunächst für MS-DOS-Rechner zur Verfügung. Die VAX-Implementierung folgt in Kürze.
Dieselbe Firma kündigte einen symbolischen Debugger auf Hochsprachenebene für C an. Das Programm wird zunächst für MS-DOS-Rechner und in Kürze auch für die VAX-Umgebung unter VMS zur Verfügung stehen.
Der Debugger benutzt für die Anzeige und die Ausführung der Debugging-Funktionen Struktogramm- und Source-Files, die vom C-Compiler-Frontend des Software-Entwicklungspaketes X-Tools übersetzt wurden. Im Kommando-Modus ist es möglich, sich die Variablenwerte lokaler und globaler Symbole anzuschauen und Ausdrücke auf Hochsprachenebene ausführen zu lassen. Die Interpretationsanzeige erfolgt in einem Fenster, in dem Struktogramm- oder Source-Files angezeigt werden. Zu den wesentlichen Eigenschaften des X-Tools-Debuggers gehören die Funktionen Breakpoint, Watchpoint und Tracepoint.

Mit der Festplatte telefonieren

Wie oft passiert es, daß man wichtige Nachrichten nicht telefonisch durchgeben kann, weil der Empfänger gerade in einer Besprechung, unterwegs oder einfach nicht da ist. Diesen nutzlosen Anrufen will die Firma Telenorma ein Ende machen. Sie hat ein digitales Sprachspeichersystem entwickelt, das gesprochene Nachrichten speichert. Das System ist eine Erweiterung von Telenormas ISDN-Anlage „Integral".
Die für ihn hinterlassenen Nachrichten kann der Teilnehmer jederzeit, von einem beliebigen Telefon aus, abhören. Zugang verschafft ihm sein persönliches Paßwort. Der digitale Sprachspeicher ist auch in der Lage, Diktate entgegenzunehmen, vertrauliche Nachrichten zu verwahren, Informationen zu vervielfältigen, Termine zu überwachen oder Nachrichten zu verteilen. Alle wesentlichen Bedienschritte gibt die Anlage dem Benutzer durch Ansagen vor. Die am häufigsten benötigten Funktionen wählt man durch Tastendruck. Einzige Voraussetzung für die volle Nutzung der Sprachspeicherfunktion ist ein Telefon mit Mehr-Frequenz-Wählverfahren (MFV) oder ein MF-Tonsender zum Erzeugen der Steuersignale. Als Speichermedium dient eine Festplatte. Sie kann Texte bis zu einer Dauer von mehreren Stunden aufnehmen.



Über ein Bildschirmterminal wird das Sprachspeichersystem von Telenorma bedient

Atari senkt die Preise

Einige Wochen bevor die Mega ST-Modelle von Atari verfügbar sind, hat das Unternehmen die Preise für die Modelle 1040 ST und 520 STM (M steht für Fernseh-Modulator) gesenkt. Das bisherige Spitzenmodell 1040 ST kostet jetzt inklusive monochromem Monitor 1698 DM und damit 300 DM weniger als bisher; der 520 STM ohne Monitor und Maus kostet jetzt 598 DM. Die Mega ST-Modelle wird es ab Anfang August in einer 2- (etwa 3000 DM) und einer 4-MByte Version (etwa 4000 DM) geben. Auf eine Version mit 1 MByte RAM wurde verzichtet. Die Mega ST-Modelle werden mit einem neuen ROM-TOS, das Routinen für den Bit-Blit-Chip (Blitter) enthält, und dem Blitter ausgestattet sein. Das neue ROM-TOS soll ohne Kompatibilitätsprobleme auch in den anderen ST-Modellen eingesetzt werden können. Ob und wie in diese Computer auch der Blitter, der die Geschwindigkeit der Bildschirmdarstellung unter GEM beschleunigen soll, auch nachgerüstet werden kann, ist noch nicht bekannt.

Die Mega ST-Modelle von Atari gibt es mit dem Blitter in einer 2- und einer 4-MByte-Version

Computer im Verbund

Ein neues Konzept für die Arbeit in einem heterogenen Computernetz gibt es von Apollo Domain: Das Network Computing System (NCS).
Innerhalb dieses Systems können nicht nur, wie in konventionellen Netzwerken, auf Dateien von Fremdrechnern zugegriffen, sondern auch nicht ausgelastete Computer von anderen Computern ausgenutzt werden. Mit der verteilten Rechenleistung werden vernetzte, heterogene Computersysteme in ihrer Gesamtheit wesentlich produktiver genutzt, da spezialisierte Rechner-Architekturen, wie z. B. Vektorrechner, KI-Maschinen und Grafik-Systeme, nicht mehr nur einem Benutzer, sondern allen mit NCS arbeitenden Systemen im Netzwerk zur Verfügung stehen. Sogar die Kommunikation eines PCs mit einem Cray-Supercomputer ist unter NCS möglich (*Bild*).
Das Network Computing System besteht im wesentlichen aus drei Software-Komponenten. Diese lauten: Remote Procedure Call Facility (RPC), Network Interface Definition Language (NIDL) Compiler und Location Broker.

Remote Procedure Call Facility (RPC)

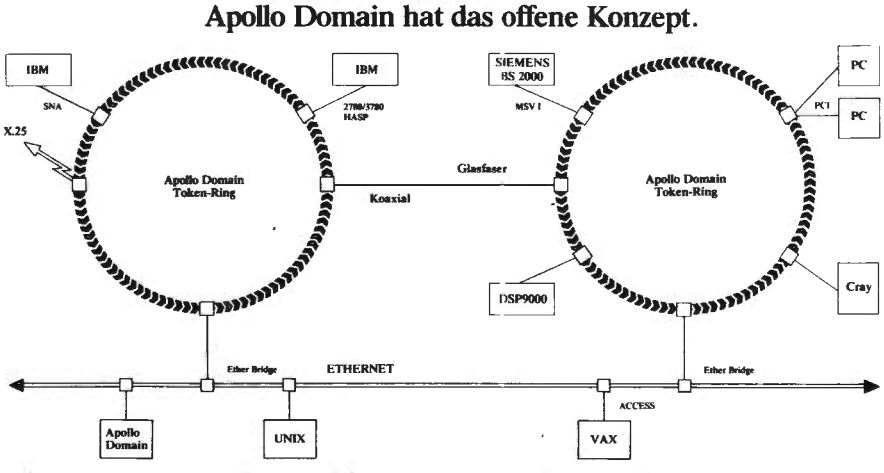
Mit dieser Softwarekomponente kann das jeweilige Anwenderprogramm Prozeduren aufrufen, die auf anderen Rechnern des Netzwerkes ausgeführt werden sollen. Die Laufzeitumgebung des RPC steuert die Aussendung und den Empfang von Daten zwischen Programmteilen auf dem Applikationsrechner und den weiteren Rechnern im Netzwerk. RPC führt die Paketierung, die Übertragung und den Empfang der Daten durch. Eventuell auftretende Fehler werden korrigiert.

Network Interface Definition Language (NIDL) Compiler

Um problemlos Daten mit RPC zu übertragen, müssen zwischen den Programmteilen, die auf der Arbeitsstation zur Bearbeitung verbleiben, und jenen Teilen, die zu Fremdrechnern geschickt werden, Programm-Zwischenstücke (stubs) erzeugt werden. Die vom NIDL Compiler erzeugten Programm-Zwischenstücke enthalten Schnittstellenbeschreibungen, so daß sich der Anwender nicht um die speziellen Eigenschaften der Rechner kümmern muß, die seine Arbeitsstation entlasten.

Location Broker

Diese Softwarekomponente stellt innerhalb des heterogenen Netzwerkes automatisch fest, welche Rechner über die erforderliche freie CPU-Leistung verfügen. Der Location Broker weist den Zielmaschinen Teilberechnungen zu. Durch Abbruch oder Fehler nicht ausgeführte Berechnungen werden erkannt und nochmals an einen freien Rechner übergeben. Der Anwender braucht in den Quelltexten seiner Programme nicht die gewünschten Zielmaschinen anzugeben.
Zahlreiche Computerhersteller unterstützen bereits das Network Computing System. Dazu gehören z. B. Siemens, Sun Microsystems, Lotus und Tandem.
Apollo Domain sieht in NCS einen entscheidenten Fortschritt in der Arbeitsleistung von Computern. Die Rechner der fünfziger und sechsziger Jahre waren batch-orientierte Maschinen, die ein Programm nach dem anderen sequentiell verarbeiteten. Seit den siebziger Jahren lassen Computer mit Betriebssystemen nach dem Zeitscheiben-Verfahren (Timesharing) ihre gleichzeitige Nutzung durch mehrere Anwender zu. Mit der Entwicklung von Arbeitsstationen in den achtziger Jahren bekam jeder Anwender einen eigenen Rechner, wobei die Vorteile des Zeitscheiben-Verfahrens durch die Vernetzung aufrecht erhalten werden konnten. Mit NCS können jetzt die Anwender auf die gesamten verteilten Computer-Ressourcen eines Netzwerkes für eine einzelne Anwendung zugreifen. Während einer Vorführung zeigte Apollo Domain, daß auf einer einzelnen Arbeitsstation, das Berechnen und Zeichnen des Apfelmännchens ungefähr 800 s dauert. Mit mehreren Arbeitsstationen im Verbund konnte diese Zeit auf den zehnten Teil reduziert werden!



Network Computing System: Nicht ausgelastete Rechner führen Prozeduren von anderen Rechnern aus

Intel bietet AT-Board an

Der große Erfolg des IBM PC/AT hat die Firma Intel jetzt dazu veranlaßt, eine AT-Mutterplatine auf den Markt zu bringen. Sie spricht damit insbesondere sogenannte OEMs an, also Hersteller von Produkten, in denen komplette Funktionseinheiten anderer Hersteller stecken.

Die Platine ist von den Einbaumaßen und der Funktion her kompatibel zum AT, leistet aber erheblich mehr als das Original. So enthält sie bereits E/A-Einheiten für die serielle und parallele Datenübertragung sowie zwei 32-Bit-Speichererweiterungs-Steckplätze. Anstelle der 80286-CPU verwendet Intel den 32-Bit-Prozessor 80386, der eine Leistungssteigerung um den Faktor 2 bis 3 bringt. Den Zugriff auf den sogenannten „Protected Mode" des 80386 läßt die Platine uneingeschränkt zu. Eine weitere Leistungssteigerung läßt sich mit dem Arithmetikprozessor 80287 erreichen, den man durch den neueren Typ 80387 ersetzen kann.

Intels AT-Platine wird mit BIOS und technischem Handbuch geliefert. Neben MS-DOS lassen sich natürlich auch andere Betriebssysteme installieren, z.B. unterschiedliche Unix-Versionen. Damit der Speicher auch für solche Anwendungen nicht zu knapp wird, läßt er sich auf insgesamt 16 MByte erweitern.

GRAF®
computer

Wir stellen vor:
Neue Produkte für NDR- und mc-CP/M-Computer. Vom Microsoft-Entwicklungspaket für DM 248,– bis zum BASIC-Interpreter für den 68020.

Einer der Vorteile unserer modularen Rechnersysteme ist deren fast unbegrenzte Ausbaumöglichkeit im Hardware- wie im Software-Bereich. Monat für Monat stellen wir neue Produkte vor, die teilweise von unseren Anwendern selbst entwickelt wurden.

Hier eine kleine Auswahl; Detailinformation entnehmen Sie bitte der neuen *LOOP 14* (der Anwenderzeitschrift für unsere Kunden). *LOOP 14* können Sie als kostenloses Probeexemplar mit nebenstehender Karte bestellen.

Alle hier aufgeführten Produkte sind sofort lieferbar. Sollten Sie uns oder unsere Rechner noch nicht kennen: Katalog (DM 10,– incl. Porto) anfordern!

Microsoft BASIC- und -Assembler-Entwicklungspaket · MBASIC-Interpreter, BASCOM-Compiler, M80 Makro-Assembler und TOOLS für 248,– DM!

Auf der Diskette befinden sich:

MBASIC	BASIC-Interpreter, V 5.21
BASCOM	BASIC-Compiler, V 5.4
M80	Makro-Assembler
L80	Linking Loader
CREF80	Cross Reference
LIB	Library Manager

... und die komplette Original-Dokumentation wird im Ringbuch (ca. 1,25 Kilo) mitgeliefert ...

Best.-Nr.	Bezeichnung	Preis DM
11059	Microsoft-Entwicklungstools 3½" 80 Spuren	248,00
11058	5¼" 80 Spuren	248,00

Neue Produkte für den NDR-Computer

RL-BASIC V 2.4

Das mächtigste BASIC für den NDR-Computer. BASIC-Interpreter unter allen 68000-CPUs. Unterstützt 68020 und Floating Point Unit, damit extrem schnell! Läuft allein sowie unter JADOS. Unterstützt Diskettenzugriff sowie Devices. Strukturierte Programmierung durch Befehle wie REPEAT ... UNTIL, WHILE ... WEND möglich. Sprungziele als Marken definierbar (GOSUB gibaus!). Kostengünstiges UPDATE möglich.

Best.-Nr.	Beschreibung	Preis DM
	RL-BASIC, Vollprodukt, auf Diskette (benötigt JADOS):	
10680	RL-BASIC auf 3½" Diskette	159,00
10679	RL-BASIC auf 5¼" Diskette	159,00
10682	RL-BASIC auf 8" Diskette	189,00

Color Look Up Tabelle CLUT

Universelle Color Look Up Tabelle. NDR-Bus und ECB-Bus. Eingang: RGB digital, z.B. TERM, ACRT oder COL256. Ausgang: RGB-Analog. Buchsen: IBM-Belegung oder 5 mal BNC. Farben: 256 gleichzeitig aus 262 144.

Best.-Nr.	Bezeichnung	Preis DM
10716	Handbuch	20,00
10715	Leiterplatte	50,00
10714	Bausatz, komplett	398,00
10713	Fertiggerät	488,00

MCOPY

Liest und schreibt IBM-Disketten mit dem NDR-Computer unter JADOS oder CP/M68K.

Best.-Nr.	Bezeichnung	Preis DM
11047	Für JADOS, 5¼" 80 Spuren	49,00
11049	Für JADOS, 3½" 80 Spuren	49,00
11046	Für CP/M68K, 5¼" 80 Spuren	49,00
11048	Für CP/M68K, 3½" 80 Spuren	49,00

ROB2

Universelle Leistungs-Karte für NDR- und ECB-Systeme. Acht Leistungsstufen, 5 ... 12 V, 1 A. Acht Digital-Eingänge. Zwei Analog-Eingänge. Besonders geeignet zur Ansteuerung von Gleichstrom-Motoren. Schnittstellen-Belegung kompatibel zu Fischer-Technic „computing"-Modellen.

Best.-Nr.	Bezeichnung	Preis DM
10632	Handbuch	20,00
10783	Leiterplatte	45,00
10631	Bausatz	228,00
10629	Fertiggerät	298,00

Weitere Produkte in LOOP 14.

NDR

Karte heute noch absenden!

NDR-Computer
mc-CP/M-Computer

GRAF ELEKTRONIK SYSTEME GMBH

Magnusstraße 13 · Postfach 1610 · 8960 Kempten (Allgäu) · Telefon: (0831) 6211

Teletex: 831804=GRAF · Telex: 17831804=GRAF · Datentelefon: (0831) 69330

Filiale Hamburg:
Ehrenbergstraße 56
2000 Hamburg 50
Telefon (040) 388151

Filiale München:
Georgenstraße 61
8000 München 40
Telefon (089) 2715858

Händler:

Jörg Korb · 1000 Berlin 30 · Telefon (030) 2627474

GMCP mbH · 2800 Bremen 33 · Telefon (0421) 211088

Schweiz: SYSTECH · CH-4104 Oberwil · Tel. (061) 3010 31

Modula-2

Modula-2/86 von Logitech für IBM-PC und Kompatible ab DOS 2.0 ist eine besonders hochstehende Implementation von Wirth's Sprachdefinition. Es wurde 1982 in der Schweiz für den professionellen Anwender entwickelt und hat sich als äusserst zuverlässig erwiesen. 10 000fach mit Erfolg eingesetzt!

Entwicklungssysteme

Umsteiger Set SFr. 233.–/DM 280.–
Compiler (mit Möglichkeit zur Erzeugung von Inline-Code für 8087), Linker, Editor, Dokumentation D und E, Pascal-To-Modula Translator.

Professional Set SFr. 530.–/DM 636.–
Wie Umsteiger Set, aber zusätzlicher Compiler, der Code für 80186, 80286 erzeugt und erst noch bis 50% schneller ist (braucht 512 k), schnellerer Linker.

Toolbox SFr. 299.–/DM 259.–
Run-Time Debugger, Post-Mortem Debugger, Decoder, Make, Version, Cross-Reference, Library-Sourcen.

ROM-Package SFr. 900.–/DM 1080.–
Speziallinker und Cross-Run-Time Debugger. Zum Erzeugen von Code für Standalone-Applikationen (ohne DOS, ohne BIOS).

Neu: Modula-2/86 auch für XENIX erhältlich!

Module/Utilities

EXE 2 LNK SFr. 140.–/DM 168.–
Linkt Assemblercode (.EXE-Files) zu Modula-2/86.

Make SFr. 82.–/DM 99.–

Window Machine SFr. 132.–/DM 159.–

Multikey-ISAM SFr. 585.–/DM 702.–
B^+-Bäume erlauben beliebige Records in weit weniger als einer Sekunde zu finden. Sorgfältig getestet – seit über einem Jahr keine Fehlermeldung! (Auch Source und Testprogramme erhältlich). Für ATARI in Vorbereitung.

High Resolution Timer SFr. 100.–/DM 120.–
Messen der Ausführungsgeschwindigkeit von Programmen ohne Zusatzhardware mit einer Auflösung von 0.84 µs.

Terra I/O SFr. 60.–/DM 72.–
Alle täglich benötigten Ein- und Ausgabeprozeduren (formatierte Strings, Binär, Hex, Dezimal ...).

File Attributes/File Date SFr. 60.–/DM 72.–
Module (linkbar) und Utility (ausführbar)

Random Generator SFr. 40.–/DM 48.–
Modul (linkbar) und Utility (ausführbar)

Auf alle obigen Produkte geben wir Ihnen eine **Rücknahmegarantie.**

Modula-2/86 BLS SFr. 125.–/DM 150.–
Das günstigste Modula-2 weit und breit! Compiler (ohne 8087), Linker, Editor, Post-Mortem-Debugger, Dokumentation D und E, nur solange Vorrat.

Nützliche Produkte

für den Programmierer, die wir bei der Softwareentwicklung täglich einsetzen. Dokumentation E, kein Support:

Fastback (5.14) SFr. 245.–/DM 294.–
20 MB Backup auf Disketten in ~10 Min.

Flash SFr. 126.–/DM 151.–
Disk Cache, beschleunigt alle Programme, die mehrmals auf die gleichen Daten zugreifen müssen – ein absoluter Hit (die Modula-2/86 Entwicklungsinstrumente werden mit 1,5 MB Extended RAM um ca. 50% schneller)!
Dazu empfehlen wir 640 kB Basisspeicher und 1–2 MB Expanded oder Extended RAM – verlangen Sie unser Angebot.

Back-To-Dos SFr. 115.–/DM 138.–
Startet DOS aus jeder beliebigen Applikation.

Tallscreen SFr. 115.–/DM 138.–
Wiederholen des letzten DOS Befehls/Pfades, rückwärtsblättern auf dem Bildschirm, schneller Batchfileeditor.

Hardware

Logimouse SFr. 250.–/DM 300.–
Seriell C7, mit Plus Package (Konfigurationen für Lotus 123, AutoCAD, Microsoft Windows, GEM .. Weitere Versionen: parallele Maus, LogiCadd, LogiPaint).

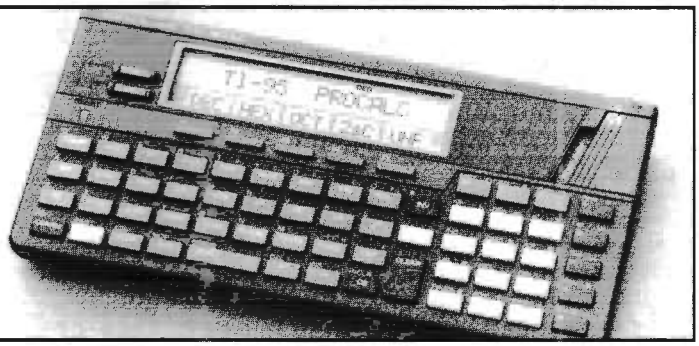
DreamLock ab SFr. 430.–/DM 516.–
das überzeugende Softwareschutzsystem auf Hardwarebasis, mit modernster Technologie. **Niemand** wird Ihre Software ohne Lizenz benutzen – jeder Fachmann ist sofort von der Sicherheit dieses Systems überzeugt.

Preise: SFr. inklusive 6,2% WUST/DM exklusive MwSt., Lieferbarkeit, Preis- und Angebotsänderungen vorbehalten.

T E R R A

TERRA Datentechnik
Stationsstrasse 62, CH-8003 Zürich, Telefon 01 462 91 71

mc-info



Fünf Funktionstasten erleichtern die Bedienung des neuen TI-95 Procalc

TI-95 Procalc ersetzt TI-59

Programmierbare technisch-wissenschaftliche Taschenrechner haben nach wie vor ihre Fans. Viele ziehen sie den in Basic programmierbaren Taschencomputern vor. Texas Instruments hat jetzt - nach langer Zeit - wieder ein solches Gerät vorgestellt, das sich bis zu 7200 Programmschritte merkt und über 200 Funktionen (darunter die Lösung kubischer Gleichungen) beherrscht. Der TI-95 Procalc führt den Bediener mit Menüs, die er auf einem zweizeiligen Display darstellt, durch die Programme. Das altbekannte „Algebraische Operationssystem" (AOS) ist einfach zu erlernen, da es der schriftlichen mathematischen Darstellungsweise entspricht. Besitzer des älteren Modells TI-59 können ihre Programme problemlos auf den neuen Rechner umschreiben.
Programme lassen sich mit Labels erstellen und assemblieren. Dies sorgt für höchste Rechengeschwindigkeit. Das interne 8-KByte-RAM ist so ausgelegt, daß es die gespeicherten Informationen auch nach dem Abschalten nicht verliert. Für Spezialgebiete sind Software-Module erhältlich, die den Funktionsumfang des Rechners wesentlich erweitern. Im Augenblick sind Module für Mathematik und Statistik verfügbar.

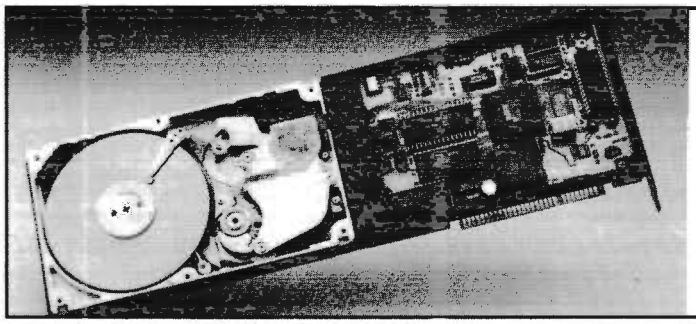
Amiga-Netz hilft Grafikern

Neue Wege bei der Höherqualifizierung arbeitsloser Grafiker hat der Verein für Fortbildung in Hannover beschritten. Er bietet Kurse an, bei denen Computer eine wesentliche Rolle spielen. Genauer gesagt: Ein Netz von zwölf Amiga-Systemen schafft die Voraussetzung, daß die Kursteilnehmer von zeitraubender Routinearbeit befreit sind und so wesentlich produktiver sein können. Ziel der Kurse ist es, den Grafikern das „Werkzeug" Computer näherzubringen.
Das Computer-Netz ist so organisiert, daß dem Kursleiter ein Amiga als Master zur Verfügung steht. An den anderen elf Geräten arbeiten die Kursteilnehmer. Unter der Zimmerdecke befindet sich ein Großbildprojektor, der den Inhalt des Bildspeichers jedes einzelnen Amigas im Raum metergroß an der Wand projizieren kann. Zwei 24-Nadel-Drucker, eine Videokamera und ein Digitalisierer vervollständigen die Ausrüstung des Computergrafik-Lernstudios. Herz der Anlage ist das Lehrerpult mit dem Amiga des Kursleiters und der Steuerung aller Geräte im Raum. Von hier aus kann der Kursleiter den Inhalt seines Bildschirms auf alle anderen Schülerschirme schalten. Er kann aber auch nur einen oder zwei damit beliefern, so daß die anderen ungestört weiterarbeiten können. Umgekehrt kann der Kursleiter das Bild eines beliebigen Monitors auf seinen Monitor holen oder in die Arbeit eines „Schülers" eingreifen.

Das neue Handwerkszeug des Grafikers: der Computer

Harddisk belegt nur einen Slot

Die meisten Festplatten, die man direkt in PCs einstecken kann, belegen einen langen Slot und ragen außerdem so in den benachbarten Steckplatzbereich hinein, daß dort nur noch eine kurze Karte Platz findet. Bei der Hardcard-40 von der amerikanischen Plus Development Corporation ist dies nicht der Fall: Nach Angaben des Herstellers ist sie die erste einsteckbare Harddisk mit 40 MByte Kapazität, die nur einen einzigen Steckplatz belegt. Das Laufwerk enthält zwei 3½-Zoll-Platten und vier Schreib-/Lese-Köpfe. Ihre mittlere Zugriffszeit beträgt 35 ms. Damit gehört sie schon zu den „Rennern" unter den PC-Plattenlaufwerken. Mit einer Leistungsaufnahme von 8 W braucht das Laufwerk keine gesonderte Stromversorgung. Der empfohlene Verkaufspreis für das Produkt beträgt 3075 DM.



Die Hardcard-40 mit geöffnetem Laufwerksgehäuse

AT unterstützt industrielle Steuerung

Der Komfort von Programmen für MS-DOS-Computer läßt sich jetzt auch in der industriellen Steuerungstechnik nutzen. Die Firma Labod Electronic hat ein Konzept entwickelt, bei dem ein AT-kompatibler Computer die Kommunikation mit dem Anwender übernimmt und gleichzeitig als Programmier- und Diagnosegerät für die Steuerung dient. Damit werden z.B. Bedienung, Meßwerterfassung und -auswertung wesentlich vereinfacht.
Die Zentraleinheit der eigentlichen Steuerung basiert auf einem 32-Bit-Mikroprozessor, den ein schneller Arithmetikprozessor unterstützt. Schnittstellenkarten für Weg- und Bahnsteuerungen bedienen bis zu acht Achsen. Eine andere Karte enthält bereits die Elektronik für die Drehzahlregelung. Sie bildet aus Soll- und Ist-Drehzahl ein analoges Signal als Steuergröße für den Motorstromregler und erfüllt daher einen Teil der Aufgaben, die bei alten Regelkonzepten im Antriebsregler erledigt werden mußten.

Ein industrieller AT erhöht den Komfort bei Steuerungsanwendungen

AST-Boards für IBM PS/2

Mit Rampage/2 und Advantage/2 hat AST die ersten Produkte für das PS/2 von IBM vorgestellt. Rampage/2 ist eine Speichererweiterung für das Modell 8530, die bis zu 2 MByte ausgebaut werden kann. Sie unterstützt den EMS- und den EEMS-Standard. Advantage/2 bietet für die Modelle 8550 und 8560 neben einer Speichererweiterung bis auf 2 MByte, die auch von OS/2 angesprochen werden kann, zusätzliche parallele und serielle Schnittstellen. Installation und Konfiguration der Steckkarten übernimmt die mitgelieferte SuperPak-Software.

Für alle Grafikkarten geeignet

Einen 15-Zoll-Multisync-Monitor mit bemerkenswerten Leistungsdaten stellt die Firma Sanyo vor. Der DMC 1537 schafft eine Auflösung von 800 Punkten in horizontaler und von 560 Punkten in vertikaler

Gestochen scharfe Farbwiedergabe erzielt man mit dem neuen Sanyo-Farbmonitor

Richtung. Der Abstand zweier Bildpunkte beträgt 0,31 mm. Das Gerät paßt sich automatisch an alle Horizontalfrequenzen zwischen 15,75 und 37 MHz sowie an die Vertikalfrequenzen zwischen 50 und 70 Hz an. Damit ist es für sämtliche gängigen PC-Grafikkarten geeignet – auch für die neue Personal-Computer-Serie von IBM. Die Bandbreite von 30 MHz sorgt für gestochen scharfe Wiedergabe. Für die einfarbige Darstellung stehen sieben Farben zur Verfügung, die der Benutzer auswählen kann. Im Preis von knapp 2000 DM sind zwei Verbindungkabel (RGB-TTL und RGB-Analog) enthalten.

Preislich jetzt noch attraktiver: der Z-159 von Zenith

Zenith senkt Preise für Z-159

Zenith hat die Preise für die verschiedenen Ausführungen des IBM PC-kompatiblen Tischcomputers Z-159, der mit einer 8088-CPU bei 8 MHz ohne Waitstates arbeitet, zum Teil erheblich gesenkt. Die Taktfrequenz kann von außen auf 4,77 MHz umgeschaltet werden. Das preiswerteste Modell mit 640 KByte RAM, Hercules- oder CGA-Grafik und zwei Disketten-Laufwerken kostet jetzt knapp 3000 DM. Dieselbe Ausführung, jedoch mit einem Disketten-Laufwerk und einer 20-MByte-Festplatte kostet 4380 DM. Der Preis des Spitzengerätes, das mit Autoswitch-EGA-Karte, einem Disketten-Laufwerk, einer 20-MByte-Festplatte und MS-Windows geliefert wird, liegt bei 6850 DM.
Bei allen genannten Versionen gehören zum Lieferumfang noch ein 12-Zoll-Monochrom-Monitor, das Betriebssystem MS-DOS 3.2 und fünf ausführliche Handbücher.

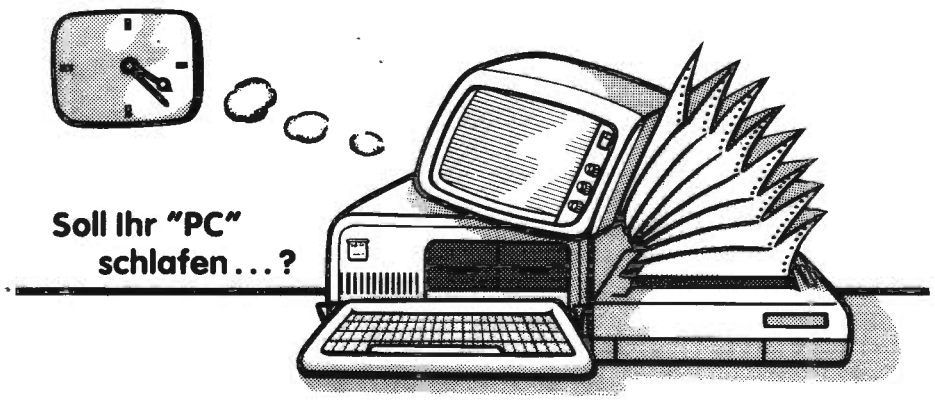
Atari-Messe

Auf dem Düsseldorfer Messegelände findet vom 18. bis zum 20. September 1987 die erste Atari-Messe statt. Softwarehäuser und Peripherie-Anbieter aus dem In- und Ausland werden ihr umfangreiches Angebot zeigen. In einem separaten Raum werden Techniker und Entwickler ihre Produkte vorstellen und mit den Besuchern diskutieren. Das Ausstellungssegment „Treffpunkt MIDI" ist den Musikfreunden gewidmet.

Desktop-Publishing: Video klärt auf

Eines der heißesten Themen der Personal-Computer-Branche ist derzeit das sogenannte Desktop Publishing (DTP). Gemeint ist der Einsatz von PCs zur Erstellung druckreifer Vorlagen mit Text und Grafik. Was es damit auf sich hat, welche Möglichkeiten, Einsatzgebiete und Grenzen sich für den PC-Anwender damit ergeben, ist das Thema eines Video-Films, den die Würzburger Firma Software Design gedreht hat.

Der Anbieter sieht darin eine Alternative einerseits zum herkömmlichen Fachbuch mit seinen begrenzten Darstellungsmöglichkeiten und andererseits zum Besuch von Seminaren oder Ausstellungen. Der Film dauert rund 120 Minuten und wird auf handelsüblichen Video-Kassetten für 210 DM angeboten.



QNX ... Ihre Lösung!

Multiuser	10 ser. Terminal pro PC, AT.
Multitasking	40 (64) Tasks pro PC (AT).
QNX-Netzwerk	2.5 MBit Tokenring. 255 PC's und/oder AT's im Netz.
Echtzeit	2800 Task Switches/sek. (AT).
Message Passing	Schnelle Intertaskkommunikation zwischen jedem Task in jedem PC.
C Compiler	K & R Standart plus Unix-Utilitys.
Flexibel	PC, Netzwerk PC, PC mit Terminals, Netzwerk PC mit Terminals. Kein Zentraler Server. Voller Zugriff auf jegliche Peripherie.
PC-DOS	Läuft als QNX-Task.
Software zu QNX	Von Datenbank über Mailbox Kalkulation bis Textverarbeitung.
Kosten	ab DM 1.226,- Demodisk gegen DM 10,- (AT oder PC)

QNX, PC-DOS und UNIX sind eingetragene Warenzeichen

INGENIEURCONSULT NL SCHWEERS GMBH
D-4005 MEERBUSCH 2 · KAPELLENSTR. 57 · TEL. 0 21 59 / 40 91 · TX. 8 531 066 · DXP 45215910168 · MODEM 50595

Ihre Lösung... CONSULT

Praktische Einführung in Lisp

E. D. Schmitter: Lisp, die Sprache der künstlichen Intelligenz. Grundlegende Eigenschaften, Algorithmen, Programmbeispiele. 218 Seiten, kart., 29.80 DM. Hofacker Verlag, Holzkirchen, 1987.
ISBN 3-88963-229-7

Der Autor des vorliegenden Bandes versteht es meiner Meinung nach sehr gut, dem Anfänger die abstrakten Quälereien zu ersparen, die andere Lisp-Bücher auszeichnen. Denn anstelle von philosophischen Bemerkungen über gepunktete Paare, Listenstrukturen und Lambda-Kalkül wird hier ganz pragmatisch an die ersten Listen ohne Vorrede herangegangen und dann frischweg immer praktisch mit dem XLisp-Interpreter (Public Domain Software) gearbeitet. Naturgemäß sind die ersten Dinge nur kleinste Demonstratiönchen. Aber nur so wird klar, wann Lisp was auswertet und weshalb Lisp eine so besondere Sprache ist. Die Übungsbeispiele sind gerade so ausgesucht, daß sie Schwierigkeiten gut bloßlegen und verstehen helfen. Allerdings wendet sich das Buch dennoch an abstrakte Geister, denn es werden sowohl die Prinzipien von Eliza-Programmen vorgeführt, als auch lernfähige Programme. Zum Schluß geht es mit der Konstruktion eines Prolog-Interpreters an das Eingemachte. Daß es die richtige Methode ist, Begriffe wie Klausen oder Unifikation ganz praktisch einzuführen, zeigt sich hier, denn ich glaube, daß hier ohne Ballast verstehbar gemacht wird, was das Logische an Prolog ist und daß Lisp ein unübertroffen mächtiges Werkzeug für den ist, der die Grundlagen und die Grenzen des Programmierens „an sich" kennenlernen will. Ein sehr gutes Buch. *Ro*

PAL-Praxis

Der Umgang mit den modernen programmierbaren Logikbausteinen. Von Ton Zengerink. 136 Seiten mit 74 Abbildungen. Lwstr-kart. 48 DM. Franzis-Verlag, München, 1987.
ISBN 3-7723-8551-6

In elektronischen Schaltungen werden immer häufiger programmierbare Logikbausteine (PAL, FPLA, IFL, PLA, PLD) eingesetzt. Der Autor hat das Buch für alle geschrieben, die mit programmierbaren Logikbausteinen arbeiten wollen oder einfach nur an diesen Bausteinen interessiert sind. mc-Leser, die sich den mc-AllProg zulegen wollen, und noch nicht so recht wissen, wie digitale Schaltungen mit programmierbaren Logikbausteinen entworfen werden, kommen mit diesem Buch voll auf ihre Kosten. Nach der Einführung in die Architekturen der PALs, PROMs und FPLAs werden die Grundschaltungen der PALs erläutert. Eine kurze Einführung in die Schaltalgebra frischt die, nach dem Studium oder anderer Ausbildung, längst vergessenen Gesetze, wie das Kommutativ- und das Assoziativ-Gesetz, wieder auf.
Wesentlich für das erfolgreiche Arbeiten mit PALs ist der richtige Umgang mit einem PAL-Assembler. Ausführlich stellt der Autor den Entwurf von Grundschaltungen der Elektronik, wie Decodierer/Demultiplexer, Multiplexer, Schieberegister und Zähler mit dem PAL-Assembler PALASM dar. Ergänzt wird dieses Kapitel mit einer Einführung in den Entwurf von Automaten mit Zustandsdiagrammen.
Immer mehr auf dem Markt befindliche Hilfsprogramme erleichtern die Entwicklung von Automaten. Dazu gibt der Autor einige Erläuterungen, um mit diesen Programmen erfolgreich arbeiten zu können. *St*

Tools für dBase III

J. T. Cooper & Company, Tools für dBase III Plus: C-Routinen. 97 Seiten, Tools für dBase III Plus: Grafik, 90 Seiten. Markt & Technik Verlag AG, Haar, 1987, je 98 DM.
ISBN 3-89090-446-7,
ISBN 3-89090-445-9

Mit zwei Büchern, die in Zusammenarbeit mit Ashton Tate, dem Entwickler von dBase III Plus entstanden sind, startet eine Reihe, die der Übung und Vertiefung ausgewählter Software sowie der Erweiterung dessen Funktionsumfangs dienen soll. Beide Neuerscheinungen bieten hauptsächlich zusätzliche Tools zu dem bekannten und weit verbreiteten Datenbanksystem. Die sehr knapp gehaltenen Bücher sind auf den Umgang und die Anwendung der auf der jeweils beiliegenden Diskette enthaltenen Routinen ausgelegt. Ein Referenzteil gibt eine übersichtliche und gut erklärte Zusammenstellung der einzelnen Funktionen. Der erste Band enthält 45 finanzmathematische, statistische und trigonometrische Funktionen sowie Kommandos zur Behandlung von Matrizen. Damit können dBase-Anwender Tilgungspläne aufstellen, Rentabilitäts-, Mittelwert- und Streuungsberechnungen durchführen. Der Funktionsumfang von dBase kann durch eine Schnittstelle zur Einbindung von C-Routinen weiter ausgebaut werden. Der zweite Band stellt eine Erweiterung dieses Konzeptes dar und liefert Routinen, die zur Erzeugung verschiedener Diagrammarten dienen. Durch diesen Band wird Grafik in dBase einbezogen.
Insgesamt gesehen stellen beide Bände wichtige Anregungen zum noch breiteren Einsatz dieses Datenbanksystems vor. *Dr. Theo Weller*

Maschinensprache des IBM-PC

Isa Brors, Maschinensprache des IBM-PC in der Praxis. Kart., 278 Seiten, 48 DM, Dr. Alfred Hüthig Verlag, Heidelberg, 1986.
ISBN 3-7785-1336-2

Personal Computer mit der 8088-CPU werden zwar immer preiswerter, aber nicht schneller. Wer seine Programme so schnell wie nur möglich ablaufen lassen möchte, kommt um die Programmierung in Assembler nicht umhin. Isa Brors hat sich darangemacht zu zeigen, wie Probleme analysiert, strukturiert und in Assembler programmiert gelöst werden. Nach der unvermeidlichen alphabetischen Aufzählung aller Befehle, die in sechs Befehlsgruppen unterteilt sind, wird die richtige Bedienung des Makroassemblers beschrieben, die an zahlreichen Beispielen nachvollzogen werden kann. Zu den umfangreichen Beispielprogrammen gehört in diesem Buch neben dem eigentlichen Assemblerlisting immer eine grafische Aufbereitung der Problemstellung sowie ein Ablaufdiagramm. Durch die ausgewogene Auswahl der Programme wird jeder Bereich des PC berücksichtigt. Neben der Programmierung der Hercules-Karte findet man Beispiele zu Tastatur- und Arithmetik-Routinen. An zwei Hardware-Erweiterungen (Parallel-Port und MIDI-Interface) wird gezeigt, wie eine Verbindung vom Computer zur Außenwelt hergestellt werden kann.
Da auch die erforderlichen Grundlagen zur Assemblerprogrammierung behandelt werden, ist dieses Buch für Leser mit geringen Vorkenntnissen geeignet. Leider wird nicht gezeigt, wie man von Hochsprachen aus Assembler-Routinen aufruft. *Rs*

Dieter Strauß

Apple NuBus

Der Bus des Macintosh II unter der Lupe

Mit dem Macintosh II führt derzeit auch Apple ein offenes System auf dem Markt ein. Die leistungsfähige 68020-CPU findet ihren Konterpart im NuBus, einem von Texas Instruments geschaffenen prozessor-unabhängigen Busstandard, der ebenbürtig zum VME-Bus und Multibus II ist – ein interessanter Wettstreit zwischen dem offenen Macintosh und der PS/2-Serie von IBM.

Bereits der „alte" Macintosh war für Apple ein Erfolg. Für den Macintosh II hat sich Apple einiges einfallen lassen. Das Blockdiagramm (*Bild 1*) gibt einen Überblick über die Systemarchitektur.

Für eine hohe Verarbeitungsgeschwindigkeit sorgt die mit 15,6672 MHz getaktete 68020-CPU. Einen großen Teil der Kommunikation über E/A-Kanäle übernehmen zwei VIA-Bausteine (Versatile Interface Adapter). Ein VIA-Baustein bedient den Apple Desk Bus (ADB), der die Schnittstelle zur Tastatur und Maus darstellt. Mit dem anderen VIA-Baustein ist die Interrupt-Sammelleitung der NuBus-Steckplätze verbunden. Für die Kommunikation mit dem Rest der Computerwelt ist der weitverbreitete Serial Communication Controller (Zilog SCC 8530) zuständig. Sowohl asynchrone als auch synchrone Datenübertragungs-Protokolle kann dieser Baustein handhaben. Alles, was mit Disketten-E/A zu tun hat, wird von einem kundenspezifischen IC, der „Integrated Woz Machine" (IWM), abgewickelt. Für den guten Ton – sogar in Stereo – sorgt der ASC-Baustein (Apple Sound Chip). Festplatten werden über einen E/A-Kanal nach dem SCSI-Standard (Small Computer Standard Interface) bedient.

Richtig interessant wird der Macintosh II jedoch erst durch den NuBus. Apple hat den größten Teil der NuBus-Spezifikationen übernommen und hat das Kind Apple NuBus getauft. Auf Abweichungen von den NuBus-Spezifikationen wird im folgenden Text jeweils hingewiesen.

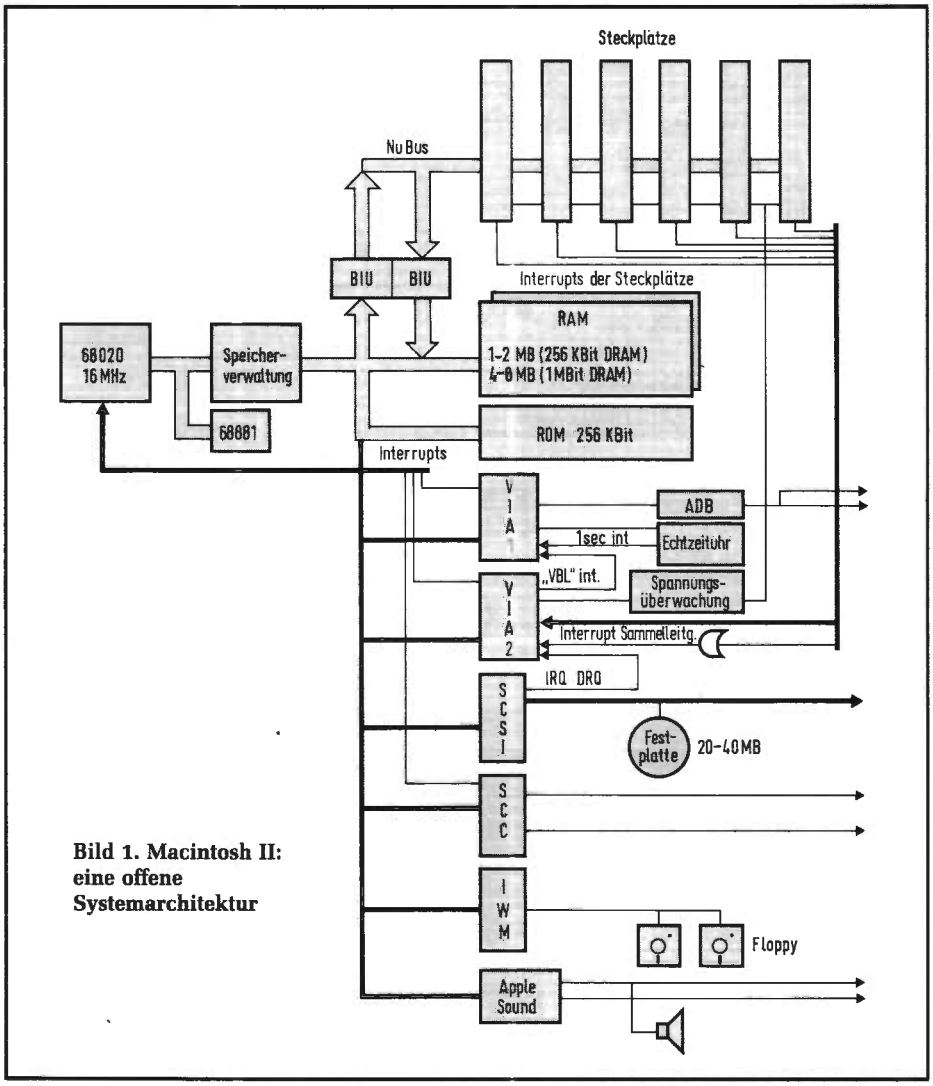


Bild 1. Macintosh II: eine offene Systemarchitektur

NuBus-Steckbrief

Der Macintosh II ist mit sechs NuBus-Steckplätzen ausgerüstet.
Der NuBus ist ein synchroner Bus, womit er sich deutlich vom VME-Bus und IBMs Mikro-Kanal-Bus unterscheidet. Alle Signalübergänge werden mit dem zentralen Systemtakt synchronisiert. Dennoch hat er viele Eigenschaften eines asynchrones Busses, da viele Buszyklen – in der NuBus-Spezifikation „transaction" genannt, in der Zahl ihrer Taktperioden variieren dürfen. Der NuBus vereint damit die leichte Anpaßbarkeit eines asynchrones Busses mit dem einfachen Entwurf eines synchronen Busses.
Die wichtigsten Eigenschaften des NuBus:

- Ein Adreßraum mit bis zu 4 GByte
- Gemultiplexter Adreß-/Datenbus (32 Bit)
- Multiprozessorfähig
- Geographische Adressierung
- Synchroner Bus mit 10 MHz-Takt
- 96polige indirekte Steckverbinder (DIN 41612, Baureihe C)
- Dezentrale Buszuteilung (Arbitration)

Die NuBus-Signale sind in fünf Klassen unterteilt:
Hilfs-, Steuer-, Adreß-/Daten-, Arbitration- und Paritäts-Signale (*Bild 2*).

Klassifizierung	Signal			Zahl der Pins
Hilfssignale	RESET			1
	CLK			1
	ID0...ID3	Identity		4
	PFW	Power Fail Warning		1
	NMRQ	Non Master Request		1
Steuersignale	START	Transfer Start		1
	ACK	Transfer Acknowledge		1
	TM0	Transfer Mode 0		1
	TM1	Transfer Mode 1		1
Adressen/Daten	AD31...AD0	Address-/Data		32
Buszuteilung	ARB3...ARB0	Arbitration		4
	RQST	Bus Request		1
Parität	SP	System Parity	*	1
	SPV	System Parity Valid	*	1

* nicht im Macintosh II verwendet

Bild 2. Die NuBus-Signale lassen sich in fünf Klassen unterteilen

Zusätzlich gibt es noch die Stromversorgungsleitungen: +12 V, +5 V, −5,2 V und −12 V.
Im Apple NuBus werden die Paritäts-Signale und und die Leitung mit −5,2 V nicht verwendet.

Der NuBus läßt nur Lese- und Schreibzugriffe in einem einzigen Adreßraum zu, d. h. ein besonderer E/A-Adreßraum wie z. B. beim PC-Bus steht nicht zur Verfügung. Jeder der sechs NuBus-Steckplätze des Macintosh II ist mit einem festverdrahteten Identifizierungscode, kurz ID genannt, belegt. Die oberen 256 MByte des 4 GByte großen Adreßraumes werden in der NuBus-Terminologie Slot Space genannt. Der restliche Speicherplatz ist nicht reserviert. Adressen auf NuBus-Karten haben das Format $FSXX XXXX. Der Identifizierungscode eines Steckplatzes bestimmt den Wert von S. Damit ist die Adresse einer Karte abhängig vom verwendeten Steckplatz, oder anders ausgedrückt von der „geographischen Lage" des Steckplatzes (geographische Adressierung). Das System kann die verschiedenen Karten voneinander unterscheiden, so daß in den meisten Fällen Brücken und DIP-Schalter auf NuBus-Karte überflüssig werden. Böse Zungen behaupten, daß an den unzähligen Brücken und DIP-Schaltern konventioneller Systeme schon mehr Anwender verzweifelt sind als an der eigentlichen Bedienung des Computers.
Ein belegter Steckplatz wird vom System an der Anwesenheit des Konfigurations-ROM erkannt. Um verschiedene ROM-Größen zuzulassen, ist das Konfigurations-ROM mit seiner Endadresse und nicht mit seiner Startadresse spezifiziert. Das letzte Byte dieses ROMs liegt auf Adresse FSFF FFFC. Aus den Daten aller Konfigurations-ROMs kann der NuBus Slot Manager die Typen der eingesteckten NuBus-Karten erkennen und die System-Software automatisch konfigurieren.

Hilfs-Signale

In der NuBus-Terminologie wird die Karte, die eine Adresse einer anderen Karte auf den Bus legt, als Master bezeichnet. Die so angesprochene Karte wird Slave genannt. Laut der NuBus-Spezifikation sind auch Karten zugelassen, die nur als Slave arbeiten können, da ihnen die Buszuteilungslogik fehlt.
CLK: Das Takt-Signal synchronisiert die Buszuteilungslogik und den Datentransfer zwischen den verschiedenen Karten. Die Taktfrequenz beträgt 10 MHz und das Puls-Pausen-Verhältnis ist 3:1. Signalübergänge finden im allgemeinen während der steigenden Flanke von CLK statt. Übernommen werden Signale mit der fallenden Flanke.
RESET: Dieses Signal (Leitung mit offenen Kollektor-Treibern) erfüllt, abhängig von der Signaldauer, zwei verschiedene Funktionen. Ist RESET nur eine Taktperiode lang aktiv, werden die NuBus-Schnittstellen aller Karten zurückgesetzt (Bus Reset). Um das gesamte System zurückzusetzen muß RESET mindestens 1 ms aktiv sein.

NuBus-Wort			
NuBus-Halbwort 1		NuBus-Halbwort 2	
Byte 3	Byte 2	Byte 1	Byte 0

Bild 3. Das NuBus-Wort ist 32 Bit lang

ID0...ID3: Signale zum Identifizieren des Steckplatzes einer Karte. Beim Steckplatz mit der höchsten Nummer ($F) sind die vier Signale auf Null. Die Systemplatine des Macintosh II hat den Idenitifizierungscode $0, die sechs Erweiterungs-Steckplätze liegen im ID-Bereich $9...$E. Die IDx-Signale werden zum Zuweisen eines Teils des Adreßraumes an die einzelnen Karten und von der Buszuteilungslogik verwendet.

TM1	TM0	AD1	AD0	Zyklustyp
L	L	L	L	Byte 3 schreiben
L	L	L	H	Byte 2 schreiben
L	L	H	L	Byte 1 schreiben
L	L	H	H	Byte 0 schreiben
L	H	L	L	Halbwort 1 schreiben
L	H	L	H	Block schreiben *
L	H	H	L	Halbwort 0 schreiben
L	H	H	H	Wort schreiben
H	L	L	L	Byte 3 lesen
H	L	L	H	Byte 2 lesen
H	L	H	L	Byte 1 lesen
H	L	H	H	Byte 0 lesen
H	H	L	L	Halbwort 1 lesen
H	H	L	H	Block lesen *
H	H	H	L	Halbwort 0 lesen
H	H	H	H	Wort lesen

* nicht im Macintosh II verwendet

Bild 4. Der Apple NuBus kennt 14 verschiedene Buszyklen

$\overline{PFW}$: Power Fail Warning ist ein asynchrones Signal und warnt vor einem Stromausfall. Im Macintosh II wird mit diesem Signal das Netzteil gesteuert. Mit $\overline{PFW} = 1$ wird das System eingeschaltet, mit $\overline{PFW} = 0$ wird es ausgeschaltet.
$\overline{NMRQ}$: Non Master Request ist ein zu $\overline{CLK}$ asynchrones Signal und wird als Interupt-Leitung von Slave-Karten benutzt.

Steuersignale

$\overline{START}$: Transfer Start wird vom aktuellen Bus Master zum Beginn eines Buszyklusses eine Taktperiode lang aktiviert. $\overline{START}$ signalisiert den Bus Slaves, daß auf den Adreß-/Datenleitungen eine gültige Adresse ansteht.

$\overline{ACK}$: Transfer Acknowledge ist das Quittungssignal der adressierten Slave-Baugruppe für das Ende des Buszyklusses. Genau wie $\overline{START}$ ist es eine Taktperiode lang aktiv.
$\overline{TM0}$ und $\overline{TM1}$: Der NuBus kann verschiedene Datengrößen übertragen. Ein Wort ist eine 32-Bit-Einheit, ein Halbwort ist 16 Bit lang. *Bild 3* stellt die Busbelegung der verschiedenen Datengrößen dar. Die Signale $\overline{TM0}$, $\overline{TM1}$ (Tranfer Mode), $\overline{AD0}$ und $\overline{AD1}$ codieren 16 mögliche Datenübertragungsarten (*Bild 4*). Block-Schreib- oder Block-Lese-Operationen kann der Apple NuBus nicht durchführen.

Während eine Slave-Karte das Ende eines Lese- oder Schreibzylus mit $\overline{ACK}$ quittiert, legt sie gleichzeitig auf die $\overline{TMx}$-Leitungen den Zustands-Code (*Bild 5*). Diese Information wird von der Master-Karte ausgewertet. Auch bei einem Fehler endet der Buszyklus normal und die Master-Karte muß entsprechend auf den Fehler reagieren. Wenn die Master-Karte nicht innerhalb 256 Taktperioden nach dem $\overline{START}$-Signal, das Quittungssignal ($\overline{ACK}$) erhält, wird von der Busüberwachungslogik des Macintosh II der Fehler „Bus Timeout" gemeldet. Um zu vermeiden, daß bei langsamen Slave-Karten die Busüberwachungslogik diesen Fehler meldet, kann die Slave-Karte den Zustands-Code „Zyklus später wiederholen" (Try again later) senden. Nach 256 Taktperioden versucht die Master-Karte den nächsten Zugriff.

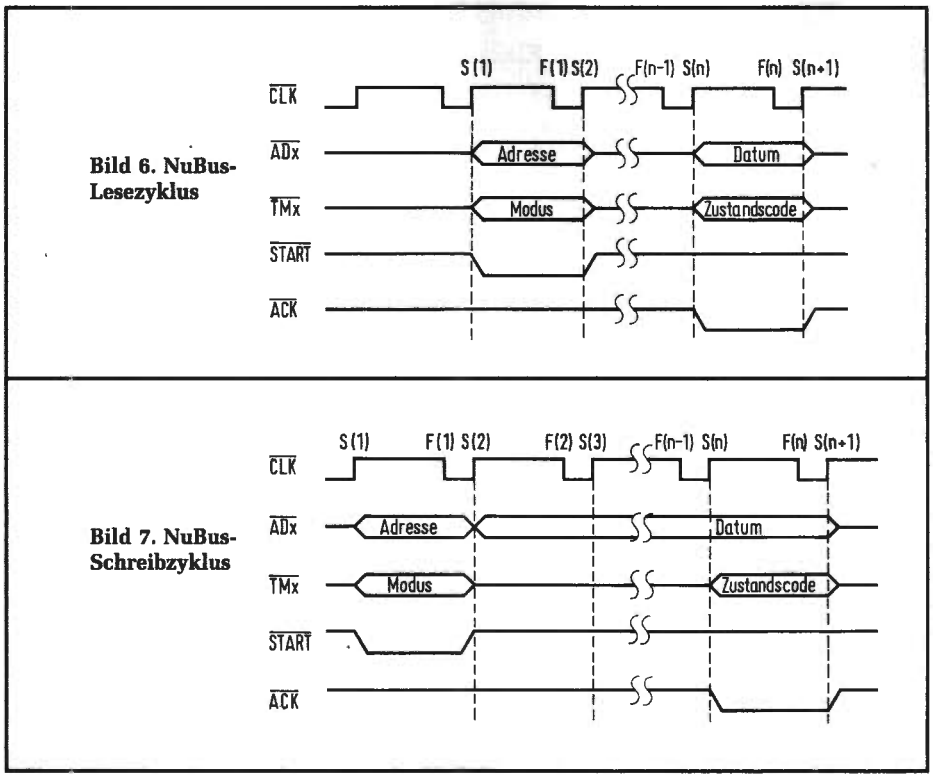


Bild 6. NuBus-Lesezyklus

Bild 7. NuBus-Schreibzyklus

Die Buszuteilungslogik

Die Buszuteilungslogik des NuBus ist fair, d. h. sie verhindert, daß höherpriore Karten ständig den Bus belegen und niederpriore Karten „verhungern" lassen. Insgesamt fünf Signale werden für die Arbitration gebraucht: $\overline{RQST}$ und $\overline{ARB0}$...$\overline{ARB3}$.
$\overline{RQST}$ (Bus Request): Karten, die einen Buszugriff vornehmen wollen, müssen zuerst die $\overline{RQST}$-Leitung (Leitung mit offenen Kollektor-Treibern) auf Low-Pegel ziehen. Wesentlich für die Funktion der NuBus-Arbitrierung ist, daß der Bus nur angefordert werden darf, wenn das $\overline{RQST}$-Signal nicht bereits aktiviert ist. Alle Karten, die gleichzeitig $\overline{RQST}$ aktiviert haben, legen ihren Identifizierungscode auf die $\overline{ARB0}$-...$\overline{ARB3}$-Leitungen. Die auf allen Karten vorhandene Buszuteilungslogik (dezentrale Arbitration!), bestimmt die Karte mit der höchsten Priorität. Nach zwei Taktperioden ist diese Karte festgestellt und sie kann mit dem Buszugriff beginnen. Wenn ihr Buszugriff beendet ist, kommt die Karte mit der jeweils niedrigeren Priorität an die Reihe. Erst wenn die letzte Karte, die gleichzeitig mit anderen Karten den Bus angefordert hat, mit ihren Transaktionen fertig ist, gibt sie die $\overline{RQST}$-Leitung wieder frei.
Nachdem die Arbitrierungslogik einer Karte den Bus zugewiesen hat, kann diese einen Lese- oder Schreibzyklus einleiten.
Um die Flanken des Taktsignals in *Bild 6* und *Bild 7* zu markieren, nennen wir eine ansteigende Flanke S(x) und eine fallende Flanke F(x); x gibt die Zahl der Taktperioden vom Beginn des Buszyklus an.

Ein NuBus-Lesezyklus

S(1): Der Bus-Master zieht die $\overline{START}$-Leitung auf Low-Pegel und legt die Adresse und den Übertragungsmodus ($\overline{ADx}$ und $\overline{TMx}$) auf den Bus.
F(1): Alle anderen Karten übernehmen mit der fallenden Taktflanke die Zustände auf den $\overline{ADx}$- und $\overline{TMx}$-Leitungen und vergleichen die Adresse mit ihrem jeweiligen Adreßbereich.
S(2): Der Bus-Master gibt die $\overline{ADx}$-, $\overline{TMx}$- und $\overline{START}$-Leitungen wieder frei und wartet auf das Quittungssignal ($\overline{ACK}$) vom Bus-Slave.
S(n): Der Bus-Slave legt das Datum auf die $\overline{ADx}$-Leitungen, quittiert mit dem

$\overline{TM1}$	$\overline{TM0}$	Zustands-Code
L	L	Zyklus beendet
L	H	Fehler
H	L	Fehler: Bus Timeout
H	H	Zyklus später wiederholen

Bild 5. Die $\overline{TMx}$-Leitungen signalisieren vier Zustände

ACK-Signal und gibt den entsprechenden Zustands-Code auf die TMx-Leitungen.
F(n): Der Bus-Master übernimmt mit der fallenden Flanke des Taktsignals das Datum und den Zustands-Code. Hat der Zustands-Code den Wert 0, ist die Übertragung erfolgreich beendet.
S(n+1): Der Bus-Slave gibt mit der steigenden Flanke des Taktsignals die ADx-, ACK- und TMx-Leitungen wieder frei. Ein neuer Lese- oder Schreibzyklus kann beginnen.

Ein NuBus-Schreibzyklus

Die ersten beiden Abschnitte eines Schreibzyklus laufen wie bei einem Lesezyklus ab. Lediglich der Zustand der TMx-Leitungen zeigt an, daß es sich um einen Schreibzyklus handelt.
S(2): Der Bus-Master legt das Datum auf die ADx-Leitungen, gibt die START- und TMx-Leitungen frei und wartet auf das Quittungssignal (ACK).
S(n): Der Bus-Slave quittiert den Empfang des Datums mit dem ACK-Signal und gibt den entsprechenden Zustands-Code auf die TMx-Leitungen.
F(2)...F(n): Der Bus-Slave übernimmt das Datum.

S(n+1): Nach dem Empfang des Quittungssignals (ACK) gibt der Bus-Master die ADx-Leitungen wieder frei. Der nächste Zyklus kann beginnen.

Multiprozessor-Systeme

Obwohl Zugriffe über den NuBus meistens nicht lange andauern, ist es manchmal notwendig, für eine Karte den Bus über mehrere Zyklen zu reservieren. Die Reservierung des Busses für den Master, Bus Locking genannt, ist in Multiprozessor-Systemen bei Semaphor-Operationen notwendig. Um den Bus für alle anderen Karten zu sperren, legt der Bus-Master die RQST-Leitung auf Low-Pegel. Da er die höchste ID hat, teilt ihm die Arbitrierungslogik immer den Bus zu.
In der NuBus-Spezifikation ist noch ein anderer Mechanismus festgelegt, der für Multiprozessor-Systeme unerläßlich ist: Resource Locking. Während der Master auf eine Karte mit einer lokalen CPU zugreift, darf die lokale CPU keinen Speicherzugriff durchführen. Um dies der jeweiligen CPU zu signalisieren, legt der Master zu Beginn des Buszyklusses die START- und ACK-Signale auf Low-Pegel. In der NuBus-Terminologie wird dieser Zyklus Attention Bus Lock genannt.

Pin	Reihe a	Reihe b	Reihe c
1	-12 V	-12 V	RESET
2	Masse	Masse	Masse
3	SPV	Masse	+5 V
4	SP	+5 V	+5 V
5	TM1	+5 V	TM0
6	AD1	+5 V	AD0
7	AD3	+5 V	AD2
8	AD5	*	AD4
9	AD7	*	AD6
10	AD9	*	AD8
11	AD11	*	AD10
12	AD13	Masse	AD12
13	AD15	Masse	AD14
14	AD17	Masse	AD16
15	AD19	Masse	AD18
16	AD21	Masse	AD20
17	AD23	Masse	AD22
18	AD25	Masse	AD24
19	AD27	Masse	AD26
20	AD31	Masse	AD30
21	AD31	Masse	AD30
22	Masse	Masse	Masse
23	Masse	Masse	PFW
24	ARB1	*	ARB0
25	ARB3	*	ARB2
26	ID1	*	ID0
27	ID3	*	ID2
28	ACK	+5 V	START
29	+5 V	+5 V	+5 V
30	RQST	Masse	+5 V
31	NMRQ	Masse	Masse
32	+12 V	+12 V	CLK

* Diese Pins sind miteinander verbunden, jedoch nicht an -5,2 V gelegt

Bild 8. Die Pinbelegung der Apple-NuBus-Steckverbinder

Parken erlaubt

Sind nur wenige Karten in den NuBus-Steckplätzen, kann ein Bus Master, obwohl er das RQST-Signal wieder freigegeben hat, den Bus so lange benutzen, bis eine andere Karte das RQST-Signal aktiviert. Dieses Verhalten, Bus Parking genannt, reduziert die durchschnittliche Zeit, die eine Karte braucht, um einen Buszugriff durchzuführen, da die zwei Taktperioden zur Arbitrierung entfallen. In *Bild 8* ist die Pinbelegung des NuBus-Steckverbinders gezeigt. Die Reihe b des Steckverbinders führt nur Masse und Versorgungsspannungen. Damit ist ein geringes Überprechen von Signalen gewährleistet.

Die technischen Daten des NuBus zeigen, daß Apple mit der Wahl des NuBus zum Systembus des Macintosh II eine kluge Entscheidung getroffen hat. Dieser Bus ist dem Multibus II und dem VME-Bus ebenbürtig, wenn nicht sogar überlegen. Erste Karten für den Apple-NuBus werden bereits von Fremdherstellern angeboten.

Spruch des Monats

„Wir haben gesehen, daß Kommunikation der Mörtel der Gesellschaft ist und daß diejenigen, die sich ungestörte Aufrechterhaltung der Kommunikationswege zur Aufgabe gemacht haben, auch am meisten mit dem Fortbestehen oder dem Verfall der Kultur zu tun haben."

Norbert Wiener, „Mensch und Menschmaschine", Alfred Metzner Verlag, Frankfurt/Main, Berlin 1952, Seite 132

Christian Jäkel

„Monte Carlo" im Computer

Wie man den Zufall in den Griff bekommt

Ein wichtiges Instrument bei der Lösungen mathematischer Probleme mit numerischen Methoden sind Zufallszahlen. Wie man sie erzeugt und anwendet, stellt dieser Beitrag dar.

Nur physikalische Vorgänge sind in der Lage, echte Zufallszahlen zu erzeugen, etwa die pro Sekunde auftretenden radioaktiven Zerfälle eines Präparats, oder die mit einem Würfel erzielten Augenzahlen.
Im ersten Falle erhält man „normalverteilte" Zufallszahlen (die Zerfallsraten streuen um einen Mittelwert herum), im zweiten Falle sind alle Augenzahlen von 1 bis 6 gleich wahrscheinlich, die Zufallszahlen sind gleich verteilt.

Wie erzeugt man Zufallszahlen?

Um Zufallszahlen mit einem Computer zu erzeugen, muß man entweder einen zufallsabhängigen physikalischen Vorgang in das Programm einbeziehen, oder man muß sogenannte „Pseudozufallszahlen" berechnen, die zwar in Wahrheit einem strengen mathematischen Bildungsgesetz folgen, die sich aber wie Zufallszahlen verhalten.
Ein gängiges Verfahren aus der ersten Kategorie ist die Einbeziehung eines Zeitfaktors, etwa wie lange der Rechner auf einen Tastendruck wartet. *Bild 1* zeigt ein Programmbeispiel. Es liefert zwar nur eine einzige Zufallszahl, sie kann aber als Anfangswert für einen Pseudozufalls-Algorithmus dienen, so daß jeder Programmlauf andere Zufallszahlen liefert, obwohl stets der gleiche Algorithmus verwendet wird. Wenn der Rechner eine Systemuhr besitzt, kann natürlich auch diese verwendet werden.

```
100 REM**** PSEUDO-ZUFALLSZAHLEN 1 ****
110 Z=0
120 PRINT "LESEN SIE BITTE DIESEN TEXT"
130 PRINT "SORGFAELTIG DURCH; WENN SIE"
140 PRINT "FERTIG SIND, BETAETIGEN SIE"
150 PRINT "TASTE 'F' FUER 'FERTIG' !!!"
160 GET A$: Z=Z+1: IF A$<>"F" THEN 160
170 PRINT
180 PRINT "DIE ZUFALLSZAHL IST ";Z
```

Bild 1. Dieses kurze Programm liefert eine Zufallszahl; sie kann als Anfangswert für einen Pseudozufalls-Algorithmus dienen

```
100 REM ** PSEUDO-ZUFALLSZAHLEN 2 **
110 :
120 REST=1000 : NENNER=2401
130 ZUF=0 : Z=0
140 ZAHL=INT(10*REST/NENNER)
150 REST=10*REST-ZAHL*NENNER
160 ZUF=10*ZUF+ZAHL : Z=Z+1
170 IF Z<4 THEN 140
180 ZUF=ZUF/10000
190 PRINT ZUF
200 GOTO 130
```

Bild 2. Möglicher Pseudozufalls-Algorithmus: Die Zufallszahlen sind die Nachkommastellen einer Division mit langer Periode

```
100 REM**** PSEUDO-ZUFALLSZAHLEN 3 ****
110 :
120 A=9 : B=2 : MY=200000002 : ZUF=0.02
130 ZUF=A*MY*ZUF+B
140 ZUF=(ZUF/MY-INT(ZUF/MY))
150 PRINT ZUF
160 GOTO 130
```

Bild 3. Auch dieser Algorithmus beruht auf einer endlosen Division

Wie berechnet man Zufallszahlen?

Viele Programmiersprachen stellen eine Zufallsfunktion zur Verfügung, z. B. RND in Basic: mit der Befehlszeile

```
ZUF = RND(1)
```

wird der Variablen ZUF ein (pseudo-) zufälliger Wert zwischen 0.000000000 und 0.999999999 zugewiesen, wobei jeder Wert gleich wahrscheinlich ist, d. h. die Zufallszahlen sind gleichverteilt.
An zwei Programmbeispielen sollen Pseudozufalls-Algorithmen demonstriert werden. *Bild 2* zeigt das erste Beispiel: Als Zufallszahlen dienen die Nachkommastellen einer Division mit langer Periode, im Beispiel 1000/2401. Die Abfolge der so erzeugten Zufallszahlen wird durch den anfänglichen Divisionsrest REST in Zeile 120 festgelegt; hier läßt sich also ein zeitbestimmter Anfangswert einsetzen.
Das zweite Beispiel (*Bild 3*) beruht ebenfalls auf einer endlosen Division, jedoch wird jeweils der komplette Nachkommateil von ZUF/MY zur neuen Zufallszahl verrechnet, gemäß der Formel in Zeile 130. Durch Variieren von A und B kann der Algorithmus modifiziert werden, dabei muß B eine gerade Zahl und A um 1 größer als eine Zweierpotenz sein; im Beispiel ist A=2*2*2+1 und B=2.
Die von diesen Algorithmen erzeugten Zahlen liegen zwischen 0 und 0.999999999. Braucht man einen anderen Zahlbereich, so werden die Zahlen noch mit einem entsprechenden Faktor multipliziert, für Zahlen zwischen 0 und 48.9999999 etwa mit 49. Schneidet man den Nachkommateil ab, erhält man gleichverteilte ganze Zahlen von 0 bis 48. Mit der Basic-Zeile

```
Z=INT(49*ZUF)+1
```

entstehen somit simulierte „Lottozahlen" zwischen 1 und 49.

Wie gut sind Zufallszahlen?

Ob die von einem Algorithmus gelieferten Zufallszahlen geeignet sind, einen „echten" Zufall (Würfel) zu ersetzen, kann mit dem „Chi-Quadrat-Test" untersucht werden. Dahinter steckt folgende Überlegung:
Geht man davon aus, daß z. B. ein Würfel simuliert werden soll, so müssen die Zahlen 1 bis 6 mit gleicher Wahrschein-

lichkeit auftreten. Ds bedeutet nun allerdings nicht (wie Glücksspieler immer wieder fälschlich glauben), daß bei 60 Würfen jede Augenzahl 10mal kommen muß; die Anzahlen werden vielmehr um 10 herum streuen. Eine Augenzahl kommt vielleicht 11mal, eine 9mal, eine andere 12mal. Wenn die Zufallszahlen „gut" sind, folgt diese Streuung einem charakteristischen Gesetz, nämlich der bereits erwähnten Normalverteilung. Das bedeutet, daß kleine Abweichungen vom „Erwartungswert" 10 häufiger, größere Abweichungen seltener auftreten. Da die Abweichungen nach oben und nach unten auftreten, würden sie sich in der Summe im Mittel aufheben. Um die Gesamtgröße der Abweichungen zu erfassen, addiert man daher nicht die Abweichungen selbst, sondern ihre Quadrate. Die Größe der insgesamt auftretenden Abweichungsquadrate – prozentual bezogen auf den Erwartungswert – trägt den Namen χ^2 (Chi-Quadrat). Dieser Wert muß innerhalb gewisser Grenzen bleiben.

Das Programm gemäß *Bild 4* testet Zufallszahlen, die von dem Pseudozufalls-Algorithmus (ab Zeile 480) geliefert werden, in dieser Weise. Durch die Festlegung x=6 in Zeile 120 werden Würfelergebnisse simuliert. Die Häufigkeit der Augenzahlen wird grafisch dargestellt, in Zeile 420 und 450 wird Chi-Quadrat berechnet. *Bild 5* zeigt einen Lauf des Programms als Beispiel.

Man vergleicht nun Chi-Quadrat mit den Werten in *Bild 6*. In der Spalte „Freiheitsgrad" sucht man in diesem Fall die Zahl 5 auf: Wenn man mit einem Würfel eine Anzahl Würfe ausführt, hat man fünf Freiheitsgrade: nämlich die Anzahl der Einsen, Zweien, ...Fünfen. Die Anzahl der Sechsen ist nicht mehr frei, das müssen ja die restlichen Würfe sein. In der Zeile Freiheitsgrad = 5 findet man nun die zulässigen Werte für Chi-Quadrat. Wenn das Ergebnis kleiner als 11,07 ist, dann ist die Wahrscheinlichkeit größer als 5 %, daß ein echter Würfel ebenfalls ein solches Ergebnis geliefert hätte, zwischen 11,07 und 15,09 hätte ein echter Würfel ein solches Ergebnis noch mit 5 %...1 % Wahrscheinlichkeit geliefert. In Bild 5 war Chi-Quadrat = 3, in deutlich mehr als 5 % der Fälle hätte also auch ein echter Würfel dieses Ergebnis geliefert. Bei einer Signifikanz von 5 % gelten die Zufallszahlen als akzeptiert. Da 3 <11,07 ist, sind also unsere Zufallszahlen im Beispiel „gut".

```
100 REM***** Z U F A L L S T E S T *****
110 :
120 X=6 : REM    ZAHLENBEREICH (1 BIS X)
130 DIM A(X),W(X) : CQ=0
140 FOR I=1 TO X
150 : A(I)=0
160 NEXT I
170 PRINT : PRINT "ZUFALLSTEST"
180 PRINT "==========="
190 INPUT "WIE VIELE ZAHLEN ...";N
200 FOR L=1 TO N
210 : GOSUB 480 : REM GENERATOR AUFRUFEN
220 : PRINT Z,
230 : A(Z)=A(Z)+1
240 NEXT L
250 PRINT : PRINT "AUSWERTUNG"
260 PRINT "==========="
270 PRINT "   0    5   10   15   20"
280 PRINT "   +----+----+----+----+----"
290 FOR I=1 TO X
300 : W(I)=A(I)/N
310 : PRINT I;
320 : FOR J=0 TO A(I)
330 :   PRINT "*";
340 : NEXT J
350 : PRINT
360 NEXT I
370 PRINT "   +----+----+----+----+----"
380 PRINT "EREIGNIS  WAHRSCHEINLICHKEIT"
390 PRINT "----------------------------"
400 FOR I=1 TO X
410 : PRINT I,W(I)
420 : CQ=CQ+(A(I)-N/X)^2
430 NEXT I
440 PRINT "----------------------------"
450 CQ=CQ*X/N
460 PRINT "CHI-QUADRAT = ";CQ
470 END
480 REM ZUFALLSZAHLENGENERATOR
490 ZUF=RND(1)      : REM ZUFALLSZAHL 0-1
500 Z=INT(X*ZUF)+1 : REM ZUFALLSZAHL 1-X
510 RETURN
```

Bild 4. Dieses Programm prüft Zufallszahlen mit dem Chi-Quadrat-Test

```
ZUFALLSTEST
===========
WIE VIELE ZAHLEN ... 15

 4
 1
 2
 3
 1
 5
 2
 3
 3
 5
 6
 1
 3
 1
 6

AUSWERTUNG
==========
   0    5   10   15   20
   +----+----+----+----+----
 1 *****
 2 ***
 3 *****
 4 **
 5 ***
 6 ***
   +----+----+----+----+----
EREIGNIS  WAHRSCHEINLICHKEIT
----------------------------
 1              .266666667
 2              .133333333
 3              .266666667
 4              .0666666667
 5              .133333333
 6              .133333333
----------------------------
CHI-QUADRAT =  3

READY.
```

Bild 5. Ein solches Ergebnis liefert das Programm von Bild 4

CHI-QUADRAT		
FREIHEITS-GRAD	IRRTUMSW'KEIT 5 %	1 %
1	3.84	6.64
2	5.99	9.21
3	7.82	11.35
4	9.49	13.28
5	11.07	15.09
6	12.59	16.81
7	14.07	18.48
8	15.51	20.09
9	16.92	21.67
10	18.31	23.21
15	25.01	30.59
20	31.41	37.57
25	37.69	44.31
30	43.77	50.89
40	55.76	63.69
60	79.08	88.38
120	146.57	158.95

Bild 6. Das Ergebnis des Chi-Quadrat-Tests vergleicht man mit dieser Tabelle

Um einen Zufallszahlen-Algorithmus auf seine Brauchbarkeit hin zu testen, wird er einfach ab Zeile 480 in das Programm nach Bild 4 eingebaut.

Wie beeinflußt man die Verteilung von Zufallszahlen?

Bisher wurden nur gleichverteilte Zufallszahlen betrachtet. Nun kann es sein, daß man für eine Anwendung Zufallszahlen braucht, die ungleichmäßig verteilt sind; bei denen also bestimmte Werte häufiger auftreten als andere. Beispielsweise sollen Zahlen nahe 0 nur selten, Zahlen nahe 1 dafür häufiger auftreten. Man kann solche Zufallszahlen auch erzeugen, wenn man nur einen Algorithmus hat, der gleichverteilte Zufallszahlen liefert. Dies funktioniert folgendermaßen:

Zunächst wäre zu überlegen, was man unter der „Verteilung" von Zufallszahlen genau verstehen will. *Bild 7* stellt die Wahrscheinlichkeit von Würfelergebnissen dar. Es gibt Ergebnisse von 1 bis 6, jedes mit einer Wahrscheinlichkeit von ⅙. Bei 49 Ergebnissen hätte jedes eine

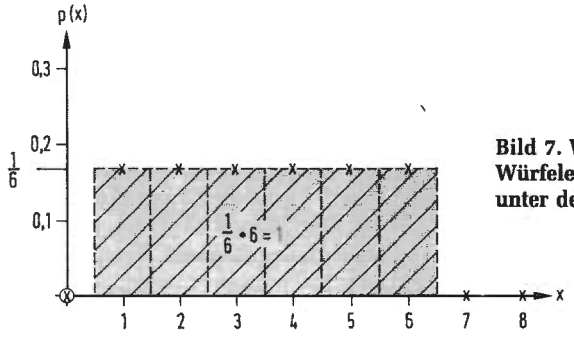


Bild 7. Wahrscheinlichkeit von Würfelergebnissen: Die Fläche unter der Kurve p(x) = 1

Wahrscheinlichkeit von 1/16, allgemein ist also der Flächeninhalt unter der Kurve p(x) gleich 1.

Bei kontinuierlich auftretenden Zahlen zwischen 0 und 1 ist die Wahrscheinlichkeit für das Auftreten einer bestimmten Zahl natürlich verschwindend, da es unendlich viele Zahlen gibt, hier betrachtet man statt dessen die Wahrscheinlichkeits-„Dichte" φ(x); damit bedeutet φ(x)Δx die Wahrscheinlichkeit, daß ein Ergebnis zwischen x und x+Δx liegt. Bei gleichverteilten Zufallszahlen ist φ(x) konstant = 1 (*Bild 8*), die Wahrscheinlichkeit, daß eine Zufallszahl zwischen 0,4 und 0,5 = 0,4 + 0,1 liegt, ist also φ(x)Δx = φ(0,4)·0,1 = 0,1. Auch hier ist die Gesamtfläche unter der Verteilungskurve = 1; man sagt, die Verteilungsfunktion sei „normiert".

Nun solle aber φ(x) linear mit x anwachsen, wie *Bild 9* es andeutet. Damit die Funktion normiert ist, muß sie die Funktionsgleichung φ(x) = 2x besitzen, denn nur dann ist die Fläche unter ihr = 1. Wenn man sich die Flächen in den einzelnen Intervallen ansieht (im Bild 9 sind vier Intervalle der Länge ¼ gewählt), so bemerkt man, daß man die Gesamtfläche in 1 + 3 + 5 + 7 = 16 gleiche Teile zerlegen kann. Von allen auftretenden Zufallszahlen dürfen dann nur 1/16 in Interfall (0...¼) liegen, 3/16 im Intervall (¼...½), 5/16 in (½...¾) und die restlichen 7/16 in (¾...1).

Wie erzeugt man solche Zufallszahlen aus gleichverteilten Zahlen? Wie *Bild 10* zeigt, muß man die gleichmäßig im Intervall (0...1) auftretenden Zufallszahlen y mit einer Funktion f(y) so umrechnen, daß das erste Sechzehntel auf das erste Viertel abgebildet wird, die nächsten 3/16 auf das zweite Viertel und so fort. Man bemerkt, daß y = f(x) der Funktionsgleichung $y = f(x) = x^2$ genügt. Da man aber nur die gleichverteilten Zahlen y zur Verfügung hat, muß man sie mit der Umkehrfunktion $x = f^{-1}(y) = \sqrt{y}$ umrechnen. Der Basic-Befehl

```
ZUF=SQR(RND(1))
```

wird also Zufallszahlen erzeugen, die der Verteilung in Bild 9 folgen.

Wie erhält man beliebig verteilte Zufallszahlen?

Allgemein beschreibt ja offenbar die Umrechnungsfunktion y = f(x) genau den Flächeninhalt unter dem Funktions-

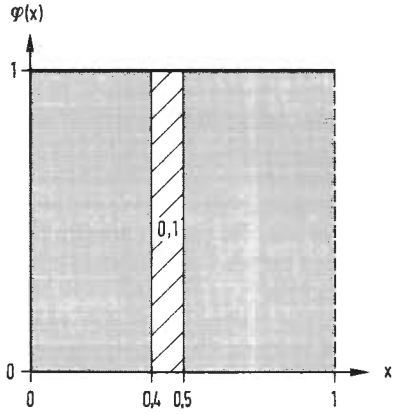


Bild 8. Bei gleichverteilten Zufallszahlen ist die Wahrscheinlichkeitsdichte konstant = 1

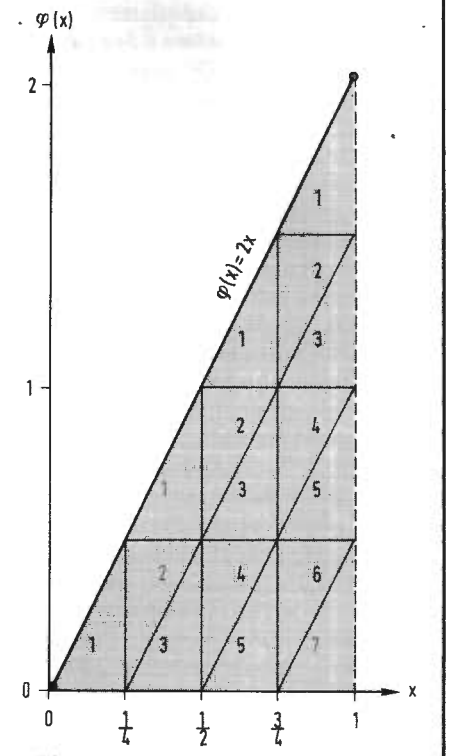


Bild 9. Die Wahrscheinlichkeitsdichte wächst hier linear mit x an

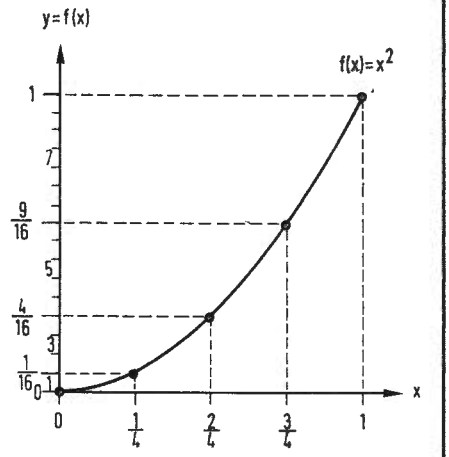


Bild 10. Die gleichmäßig auftretenden Zufallszahlen werden mit einer Funktion umgerechnet

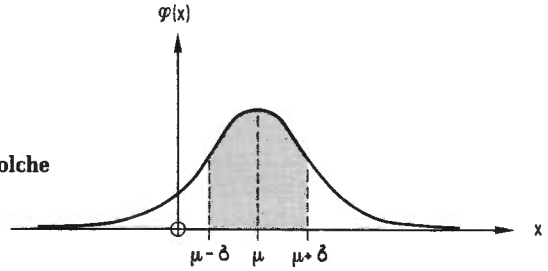


Bild 11. Normalverteilte Zufallszahlen haben eine solche Verteilungsfunktion

graphen φ(x). Demnach muß

$$y = f(x) = \int \varphi(x)\, dx$$

sein. Im vorherigen Beispiel war

$$\varphi(x) = 2x$$

also

$$y = f(x) = \int 2x\, dx = x^2 + C$$

Die Integrationskonstante muß so bestimmt werden, daß f(x) das gewünschte x-Intervall auf das y-Intervall von 0 bis 1 abbildet, denn der Zufallszahlen-Algorithmus liefert nur Zahlen in diesem Intervall. Im Beispiel war hierzu einfach C = 0 erforderlich. Wenn y = f(x) auf diese Weise bestimmt ist, bildet man die Umkehrfunktion $x = f^{-1}(y)$ und rechnet damit die gleichverteilten Zufallszahlen um.

Wie erhält man normalverteilte Zufallszahlen?

Das soeben beschriebene Verfahren ist zur Erzeugung normalverteilter Zufallszahlen leider nicht geeignet. Normalverteilte Zufallszahlen haben eine Verteilungsfunktion gemäß *Bild 11*. Der Wert mit der größten Wahrscheinlichkeit ist der Mittelwert μ (My), die Breite der Verteilung ist durch die Standardabweichung σ (Sigma) beschrieben, bei dieser Abweichung von μ ist die Wahrscheinlichkeit nur noch 60,7 % ($= 1/\sqrt{e}$) des Maximalwertes. Diese Verteilungsfunktion folgt der Gleichung

$$\varphi(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Das Integral über diese Funktion, das man ja zur Erzeugung entsprechend verteilter Zufallszahlen bestimmen müßte, kann nicht analytisch berechnet werden. Zwar ist eine numerische Berechnung mittels einer Potenzreihe möglich, jedoch hat man dann immer noch das Problem, die Umkehrfunktion bestimmen zu müssen.

Zur Erzeugung normalverteilter Zufallszahlen schlage ich daher einen anderen Weg vor: Mit dem Programm in *Bild 12* können sogenannte „Poisson-verteilte" Zufallszahlen erzeugt werden. Sie können in guter Näherung anstelle normalverteilter Zahlen verwendet werden.

In der vorliegenden Fassung erzeugt das Programm eine vorgegebene Anzahl Poisson-verteilter Zahlen, durch Weglassen der äußeren Schleife kann man es aber auch als Zufallszahlengenerator in ein anderes Programm integrieren. Neben Mittelwert und Standardabweichung muß man bei diesem Programm auch noch die Zahl der Möglichkeiten angeben, wie viele verschiedene Zufallszahlen erzeugt werden können sollen. Das Programm erzeugt dann nämlich zunächst Poisson-verteilte Zahlen im Bereich 0...M (die mit ausgedruckt werden) und rechnet diese dann in Zufallszahlen mit dem geforderten Mittelwert und der geforderten Standardabweichung um. Je größer M angegeben wird, desto besser geben die Poisson-verteilten Zahlen eine Normalverteilung wieder, allerdings steigt im gleichen Maße die Rechenzeit.

Was macht man mit Zufallszahlen?

Die Anwendung von Zufallszahlen konzentriert sich auf die Lösung mathematischer Probleme und die Simulation phy-

```
100 PRINT"ZUFALLSZAHLEN POISSONVERTEILT"
110 PRINT"============================"
120 INPUT"WIEVIEL ZAHLEN........";N
130 INPUT"WIEVIEL MOEGLICHKEITEN";M
140 INPUT"MITTELWERT BEI........";MY
150 INPUT"STANDARDABWEICHUNG....";SIGMA
160 PRINT
170 M=M-1 : S=SQR(M)/2
180 PRINT "IN 0 -";M;"/ UMGERECHNET"
190 PRINT"----------------------------"
200 FOR Z=1 TO N
210 : ZUF=0
220 : FOR I=1 TO M
230 :   ZUF=ZUF+INT(2*RND(1))
240 : NEXT I
250 : PRINT ZUF,
260 : ZUF=MY+(ZUF-M/2)*SIGMA/S
270 : PRINT ZUF
280 NEXT Z
290 END
```

Bild 12. Dieses Programm errechnet Poisson-verteilte Zufallszahlen

```
100 REM ******  PI - MONTECARLO  ******
110 :
120 CLS$=CHR$(147) : REM   CLEAR SCREEN
130 HME$=CHR$(019) : REM   CURSOR  HOME
140 R=20           : REM   KREIS-RADIUS
150 INPUT "WIE VIELE VERSUCHE ...";N
160 PRINT CLS$;
170 FOR X=1 TO R+1
180 : PRINT ":";
190 NEXT X
200 PRINT
210 PRINT HME$;
220 PRINT SPC(R+2);"PI-MONTCARLO"
230 PRINT SPC(R+2);"============"
240 PRINT HME$
250 PRINT SPC(R);"X"
260 FOR Y=1 TO R-1
270 : X=INT(SQR(R*R-Y*Y))
280 : PRINT SPC(X);"X";SPC(R-X-1);":"
290 NEXT Y
300 PRINT "X";
310 FOR X=1 TO R
320 : PRINT ":";
330 NEXT X
340 PRINT
350 T=0
360 PRINT HME$;
370 FOR Z=1 TO N
380 : GOSUB 570 : REM  GENERATOR-AUFRUF
390 : X=R*ZUF
400 : GOSUB 570 : REM  GENERATOR-AUFRUF
410 : Y=R*ZUF
420 : M$="-"
430 : IF X*X+Y*Y<R*R THEN T=T+1: M$="+"
440 : PRINT HME$;Z;
450 : FOR I=0 TO Y
460 :   PRINT
470 : NEXT I
480 : PRINT SPC(X);M$
490 NEXT Z
500 P=4*T/N
510 PRINT HME$ : PRINT
520 PRINT SPC(R+1);N;"SCHUSS"
530 PRINT SPC(R+1);T;"TREFFER"
540 PRINT SPC(R+2);"-> PI=";P
550 PRINT HME$;
560 END
570 REM ZUFALLSZAHLENGENERATOR
580 ZUF=RND(1)
590 RETURN
```

Bild 13. Dieses Programm berechnet den Inhalt eines Viertelkreises im Verhältnis zum unbeschriebenen Quadrat

sikalischer Vorgänge. Das Simulieren eines Roulette-Spiels auf dem Computer würde beispielsweise zu diesen physikalischen Vorgängen gehören, ebenso das Simulieren eines radioaktiven Zerfalls. Mit dem Programm aus Bild 12 könnte man (mit den Eingaben M = 100; MY = 50; SIGMA = 10) typische Zählraten eines Geiger-Müller-Zählrohres nachempfinden und so z. B. testen, ob ein gegebenes Gerät schnell genug reagiert, um alle eintreffenden Strahlungsteilchen zu registrieren.
Ich möchte hier aber noch zwei mathematische Anwendungen vorstellen (Wegen des Bezugs zum Roulette-Spiel tragen diese Zufallsverfahren zur mathematischen Problemlösung den Namen „Monte Carlo"). Zur Berechnung des Flächenstücks unter einer Kurve kann man – falls eine Integration nicht möglich ist – Zufallszahlen verwenden. Man schließt die Figur in ein Rechteck bekannten Inhaltes ein und belegt dieses Rechteck nun mit zufällig gewählten Punkten. Das Verhältnis der innerhalb der Kurve gelegenen Punkte (Treffer) zur Gesamtzahl der Punkte (Schuß) entspricht dem Verhältnis der Fläche unter der Kurve zur Rechtecksfläche.
Bild 13 ist ein Programm, das auf diese Weise den Inhalt eines Viertelkreises im Vergleich zum umbeschriebenen Quadrat bestimmt und den Vorgang gleichzeitig anschaulich auf dem Bildschirm darstellt. Da der Kreisinhalt πr^2 ist, hat der Viertelkreis den Inhalt $\pi r^2/4$, das Quadrat hingegen den Inhalt r^2. Das Verhältnis muß also $\pi/4$ ergeben. So eignet sich dieses Programm zur Bestimmung der Kreiszahl π, wobei die Genauigkeit mit der Anzahl der abgegebenen „Schüsse" steigt. Man benötigt für dieses Programm die Steuercodes für „Schirm löschen" und „Cursor Home", die leider von Computer zu Computer verschieden sind.
Eine zweite Anwendung gibt *Bild 14*. Dieses Programm ermittelt mit Hilfe der Monte-Carlo-Methode den geringstmöglichen Materialbedarf beim Zuschneiden von Drähten, Leisten oder dergleichen – oder beim Verteilen von Programmen gegebener Länge auf möglichst wenige Disketten.
Ausgehend von der Rohlingslänge (Länge des ungeschnittenen Materials oder Kapazität einer Diskette) und den benötigten Abschnittslängen (Programmlängen) werden fortlaufend zufällig gewählte Kombinationen der Abschnitt auf die Rohlinge verteilt, bis der Anwender einen Vorschlag akzeptiert oder die Kombination mit dem geringstmöglichen Verschnitt gefunden ist. Ein Problem, das bei größeren Materialmengen durch systematisches Durchgehen aller Möglichkeiten in vertretbarem Zeitaufwand nicht zu bewältigen wäre, während die Monte-Carlo-Methode bereits nach kurzer Zeit die ersten brauchbaren und mit wachsender Rechenzeit immer bessere Lösungen liefert.

Zitat des Monats

„Für optimale Saumfestigkeit durch Hitzeimpulse versiegelt, haben die „Charleswater CP303 Statshield® Transparent Conductive Bags" eine Resistivität von weniger als 10^5 Ohm/Quadrat an der Außenseite und weniger als 10^{12} Ohm/Quadrat an der Innenseite. Die Beutel sind ideal für schwer bestückte gedruckte Schaltungsplatten geeignet und in den Größen von 7 cm × 10 cm bis 61 cm × 61 cm erhältlich."

Aus einer Presseinformation der Charleswater Products, Ltd.
Julian Price, Managing Director, Charleswater House, Kent, England.

```
100 REM ***  ZUSCHNITT- OPTIMIERUNG  ***
110 :
120 REM ********** EINLESEN ************
130 CLS$=CHR$(147):REM CLEAR SCREEN
140 PRINT CLS$;"ZUSCHNITT-OPTIMIERUNG"
150 PRINT "== NACH MONTECARLO ==": PRINT
160 INPUT "LAENGE DER ROHLINGE  =";BL
170 PRINT : PRINT "ABSCHNITTSLAENGEN"
180 PRINT "( ENDE = 0 )"
190 S=0 : Z=0 : DIM L(1000)
200 Z=Z+1
210 PRINT "ABSCHNITT ";Z;"=";:INPUT L(Z)
220 IF L(Z)<=BL THEN 240
230 PRINT "FEHLER !" : GOTO 210
240 S=S+L(Z) : IF L(Z)>0 THEN 200
250 PRINT
260 NT=INT(S/BL+0.99999999)
270 VT=NT*BL-S : Z=Z-1 : VM=99999999
280 PRINT "MINDESTZAHL:";NT;" ROHLINGE"
290 PRINT "VERSCHNITT MINDESTENS ";VT
300 DIM B(Z,Z) : DIM M(Z,Z) : VZ=0
310 REM *********** RECHNEN ************
320 VZ=VZ+1
330 PRINT: PRINT "ES LAEUFT VERSUCH ";VZ
340 PRINT "( ABBRUCH MIT 'S' )"
350 GET A$ : IF A$="S" THEN 520
360 FOR I=1 TO Z :L(I)=ABS(L(I)): NEXT I
370 FOR I=1 TO Z : FOR J=1 TO Z
380 B(I,J) = 0   : NEXT J : NEXT I
390 B=1 : A=1 : S=0 : Q=0 : G=0
400 N=INT(Z*RND(1)+1)
410 IF L(N)<0 THEN 400
420 S=S+L(N) : G=G+L(N)
430 IF S>BL THEN S=L(N) : B=B+1 : A=1
440 B(B,A)=L(N):L(N)=-L(N):Q=Q+1:A=A+1
450 IF Q<Z THEN 400
460 V=B*BL-G : IF V>=VM THEN 320
470 VM=V
480 FOR I=1 TO Z : FOR J=1 TO Z
490 M(I,J) = B(I,J)
500 NEXT J : NEXT I
510 REM ********** AUSGEBEN ************
520 PRINT CLS$;
530 FOR I=1 TO Z
540 PRINT : PRINT "ROHLING NR. ";I
550 FOR J=1 TO Z
560 PRINT M(I,J);
570 IF J<Z THEN IF M(I,J+1)=0 THEN J=Z
580 NEXT J
590 IF I<Z THEN IF M(I+1,1)=0 THEN I=Z
600 NEXT I
610 PRINT : PRINT "VERSCHNITT IST ";V
620 PRINT " OPTIMAL WAERE ";VT
630 IF V=VT THEN 670
640 PRINT:INPUT "WEITERSUCHEN (J/N) ";A$
650 IF A$ ="J" THEN 310
660 IF A$<>"N" THEN 640
670 END
```

Bild 14. Den geringstmöglichen Materialbedarf ermittelt dieses Programm mit Hilfe der Monte-Carlo-Methode

Peter Kuhn

Das ist der IBM-Mikro-Kanal-Bus

Teil 2: Automatische Systemkonfiguration/ Arbitration/Zeitverhalten

Im ersten Teil des Artikels wurden die Signale genau geschildert, die den Micro Channel bilden. Jetzt tritt der logische Aspekt mehr in den Vordergrund. Die Komponenten eines Systems, das den Mikro-Kanal benutzt, tragen Kennzeichen, damit sie vom System identifiziert werden können.

Mit Hilfe des POS (Programmable Option Select) Verfahrens ist es möglich, sowohl die System-Leiterplatte als auch alle Adapterbaugruppen geographisch, also abhängig von der jeweiligen Kanalnummer (Steckplatz), zu adressieren. Basierend darauf kann über programmierbare Register das Gesamtsystem anhand von Software eingestellt werden, was ein umständliches Setzen von Schaltern und Steckbrücken auf den einzelnen Baugruppen erspart. Mit Hilfe von sogenannten Adapter Description Files (ADFs) und auf Diskette mitgelieferten System Konfiguration Utilities können spezielle Konfigurationsdaten für die System-Leiterplatte und die Adapter erzeugt werden. Dies wird dadurch erreicht, daß zuerst die jeweilige Adapter-Identifikationsnummer (sog. Adapter-ID, zwei Bytes) bestimmt wird. Mit Hilfe der zugeordneten Daten aus dem ADF File wird dann der jeweilige Adapter per Software konfiguriert und das Ergebnis der Konfiguration zusammen mit der jeweiligen Adapter-ID im CMOS-RAM der System-Leiterplatte gespeichert.

Beim Einschalten des Systems vergleicht nun der Rechner die Adapter-IDs der im System befindlichen Karten mit den im CMOS-RAM abgespeicherten Adapter-IDs und erkennt, ob sich die Konfiguration des Systems seit dem letzten Einschalten geändert hat. Wenn ja, dann müssen die Konfiguration Utilities neu absolviert werden, um die Daten im CMOS-RAM zu aktualisieren. Mit diesem Verfahren kann man auch komplexeste Adapter leicht integrieren.

Adresse(Hex)	$\overline{\text{CD SETUP}}$	A2	A1	A0	Funktion
0100H	0	0	0	0	POS Reg. 0 Adapter ID (LSB)
0101H	0	0	0	1	POS Reg. 1 Adapter ID (MSB)
0102H	0	0	1	0	POS Reg. 2 Optionsauswahl 1 Bit 0 = Card Enable (CDEN)
0103H	0	0	1	1	POS Reg. 3 Optionsauswahl 2
0104H	0	1	0	0	POS Reg. 4 Optionsauswahl 3
0105H	0	1	0	1	POS Reg. 5 Optionsauswahl 4 Bit 7 = CHCK Bit 6 = Reserviert
0106H	0	1	1	0	POS Reg. 6 Adresserweiterung (LSB)
0107H	0	1	1	1	POS Reg. 7 Adresserweiterung (MSB)

Bild 1. POS-I/O-Register

Die programmierbaren System-Board- und Adapter-Register belegen die Adressen 100H...107H. Sie werden durch Beschreiben der „Setup Enable" Register (94H für die Hauptplatine und 96H für alle Adapter) zur Systemkonfiguration (SETUP) oder für den normalen Betrieb (ENABLE) freigegeben. Die Register 94H und 96H sind einzig und allein nur dazu da, die Register 100H...107H entweder für die System-Leiterplatte, den Video-Grafik-Adapter (VGA) oder einen anderen Adapter freizugeben (*Bild 1*).

Die System-Initialisierung

Bild 2 zeigt die Belegung des Setup/Enable-Registers 94H auf der System-Leiterplatte. Wie man sieht, können mit diesem Register entweder Funktionen der System-Leiterplatte oder der VG-Adapter in den Setup-Modus gebracht werden. Wird der Setup-Modus für die System-Leiterplatte und für den VGA gleichzeitig eingeschaltet, dann gibt es einen Buskonflikt: Diese Einstellung ist verboten!

Bit	Funktion
7	System Board Funktionen 1 = Enable 0 = Setup
6	Reserviert
5	VGA 1 = Enable 0 = Setup
4	Reserviert
3	Reserviert
2	Reserviert
1	Reserviert
0	Reserviert

Bild 2. System Board Setup/Enable Register (94H)

Über das programmierbare Register 102H (*Bild 3*) können dann beim Setup der System-Leiterplatte mit der Software verschiedene Peripheriefunktionen eingestellt werden, die in der bisherigen PC-Architektur nur mechanisch durch Steckbrücken festgelegt werden konnten. Dabei sendet der Adapter, wenn vorhanden, bei jeder Adressierung sein Card-Select-Feedback-Signal ($\overline{\text{CD SFDBK (n)}}$) aus. Anhand dieses Signals, das im Card-Select-Feedback-Register (91H) auf der System-Leiterplatte zwischengespeichert wird, kann die Software feststellen, daß ein Adapter, der VGA oder Funktionen der System-Leiterplatte eine Adressierung erkennen können. Bit 0 des Registers 91H ist immer dann High, wenn im vorhergehenden Zyklus

Bit	Funktion
7	Parallel Port 0 = Enable bidirektionalen Mode 1 = Disable bidirektionalen Mode
6 5	Funktion / Adresse
0 0	Parallel 1 — 03BCH - 03BEH
0 1	Parallel 2 — 0378H - 037AH
1 0	Parallel 3 — 0278H - 027AH
1 1	Reserviert
4	1 = Enable Parallel Port 0 = Disable Parallel Port
3	1 = Serial Port = Serial 1 0 = Serial Port = Serial 2
2	1 = Enable Serial Port 0 = Disable Serial Port
1	1 = Enable Diskettenlaufwerk 0 = Disable Diskettenlaufwerk
0	Übergeordnetes Enable/Disable 1 = Bit 4,2,1 sind gültig in obiger Bedeutung 0 = Disable Parallel/Serial Port und Diskettenlaufwerk unabhängig von der Bedeutung der Bits 4,2,1

Bild 3. System Board I/O Byte (102H)

ein I/O-Zugriff auf diese Komponenten stattgefunden hat. Beim Lesen des Registers wird Bit 0 automatisch zurückgesetzt. Bit 1...7 sind reserviert.

Die Adapter Board Initialisierung

Bei der Initialisierung von Adapterbaugruppen ist ähnlich wie bei der Initialisierung der System-Leiterplatte vorzugehen. Zuerst muß die Software das Setup-Enable-Register 96H mit dem entsprechenden Bitmuster (*Bild 4*) belegen, um die $\overline{\text{CD-SETUP-(n)}}$-Leitung für den gewählten Adapter zu aktivieren. Durch die Hardwarelogik auf dem adressierten Adapter (*Bild 5*) wird dieser für das Beschreiben der programmierbaren Register 100H...107H vorbereitet. Der Inhalt von Register 96H kann jederzeit von der Software gelesen werden, Bit 4,5 und 6 sind beim Auslesen allerdings immer 1.

Zum Programmieren der Funktionen der System-Leiterplatte (über Register 94H) muß übrigens das Register 96H immer auf 0 gesetzt werden. Die 8-Bit-Read-Only-Register 100H und 101H enthalten die zwei Byte lange Adapter-ID für die Konfiguration-Utilities, Diagnostik- oder Initialisierungs-Routinen. Die Adapter-ID muß immmer 00 enthalten, wenn ein Adapter nicht initialisiert wurde oder nicht imstande ist zu antworten. Falls die System-Board-Software länger als eine Sekunde die Adapter-ID 00 erkennt, nimmt sie an, daß die entsprechende Baugruppe fehlerhaft ist und bringt eine entsprechende Fehlermeldung. Ansonsten wird die Adapter-ID entsprechend der Funktion des jeweiligen Adapters beschrieben (*Bild 6*). Die Read/Write-Register 102H...105H sind beliebig programmierbar – bis auf folgende Ausnahmen:

Register 102H, Bit 0 : Card Enable (CDEN)

Disable (0), Enable (1) Adapter. Wenn dieses Bit 0 ist, reagiert der Adapter nur auf Setup-Lese- und Setup-Schreiboperationen, nicht jedoch auf Speicher- und I/O-Operationen oder Interrupt Requests.

Register 105H, Bit 6,7 : Channel Check Active Indicator

Bei einem Fehler auf einer Adapterbaugruppe kann diese Bit 7 auf Low ziehen und damit die Leitung $\overline{\text{CHCK}}$ aktivieren. Da diese Leitung allen Adaptern gemeinsam ist, muß zur Lokalisierung des Feh-

Kanalnummer für Setup	7	6	5	4	3	2	1	0
Keine	0	0	0	0	0	X	X	X
1	0	0	0	0	1	0	0	0
2	0	0	0	0	1	0	0	1
3	0	0	0	0	1	0	1	0
4	0	0	0	0	1	0	1	1
5	0	0	0	0	1	1	0	0
6	0	0	0	0	1	1	0	1
7	0	0	0	0	1	1	1	0
8	0	0	0	0	1	1	1	1
Kanal Reset	1	0	0	0	X	X	X	X

Bild 4. Kanalauswahlregister (96H)

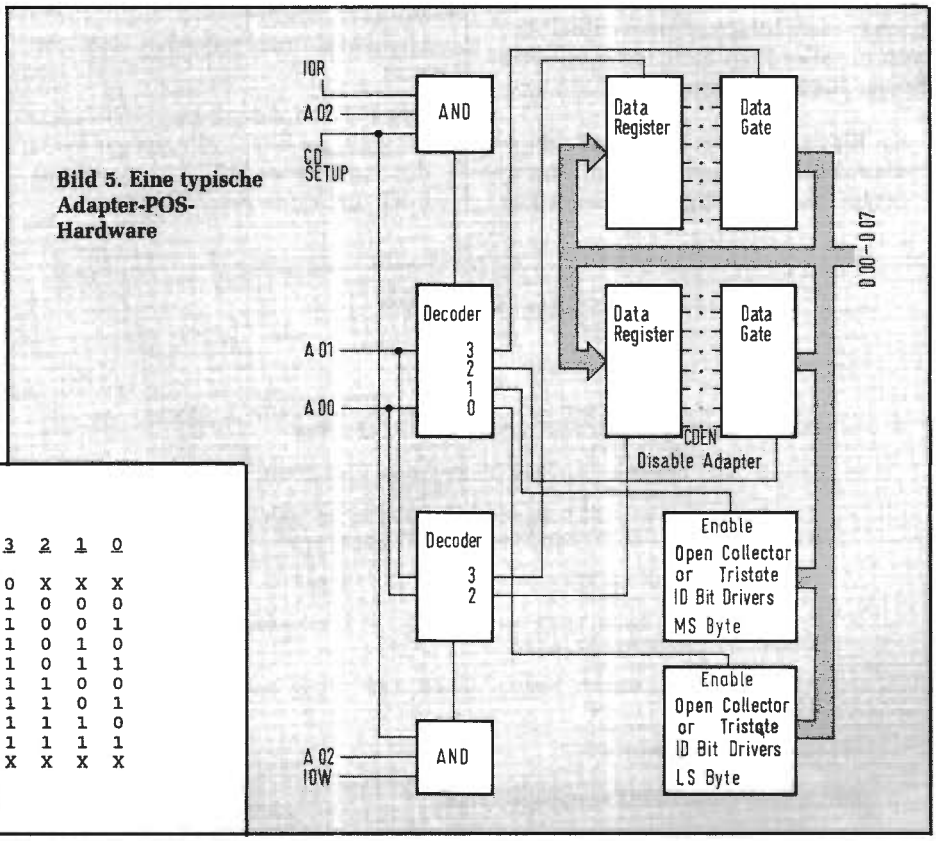


Bild 5. Eine typische Adapter-POS-Hardware

Adapter ID	Definition
0000H	Adapter nicht bereit
0001H - 0FFFH	Bus Master
5000H - 5FFFH	DMA Adapter
6000H - 6FFFH	Adapter unter direkter Programm Kontrolle z.B. I/O Adapter
7000H - 8FFFH	Speicherbaugruppe
FFFFH	Adapter nicht angesprochen

Bild 6. Adapter-ID für verschiedene Adapter

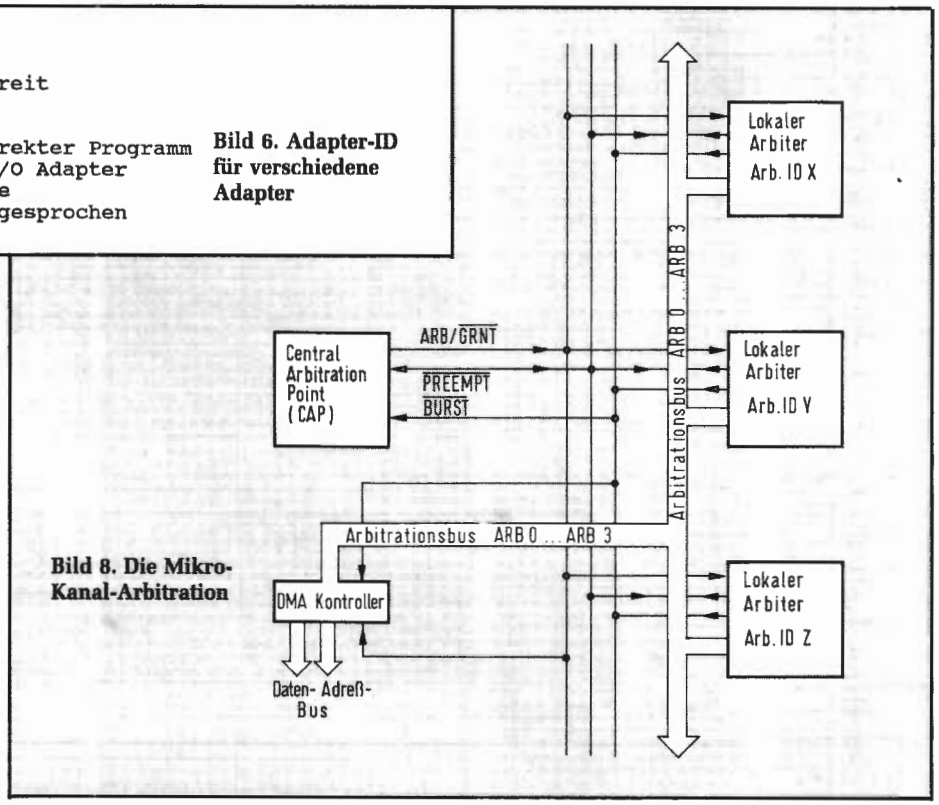


Bild 8. Die Mikro-Kanal-Arbitration

lers auf den einzelnen Adaptern Bit 7 unter der Adresse 105H gelesen werden, um festzustellen, welcher Adapter eine Fehlermeldung ausgelöst hat. Wird gleichzeitig mit Bit 7 auch Bit 6 auf Low gelegt, zeigt der Adapter an, daß ein Fehler-Code oder ein Verweis auf einen Fehler-Code in den POS-Bytes 106H und 107H abgelegt wurde. Alle Adapter, die $\overline{\text{CHCK}}$ unterstützen, müssen diese beiden Bytes liefern.

Die Register 106H und 107H (Subadress Extension) erlauben außerdem die Adressierung eines Unterblocks von Daten auf einem Adapter. Sie können zum Beispiel benutzt werden, um ein lokales RAM zu adressieren. Hierbei wird zuerst die Adresse über Register 106H und 107H eingestellt und in ein lokales Zähler-Register geladen, welches anschließend bei jedem Zugriff auf eines der POS-Register inkrementiert wird.

In den POS-Bytes 102H...105H müssen noch weitere Informationen enthalten sein, die allerdings nicht festen Bit-Positionen zugeordnet sind.

- Ein Bit muß als sog. Fairness-Feld für die Arbitration ausgelegt sein. Dies bedeutet für den lokalen Arbiter, daß er sich erst wieder um die Arbitration bewerben darf, wenn alle anstehenden Arbitrationsanforderungen der Priorität nach abgearbeitet worden sind.
- Weitere 4 Bits müssen für die Zuweisung der Arbitration-ID, der Buspriorität im Multimaster-Betrieb, reserviert sein.
- Alle I/O Adapter mit Memory Mapped I/O sollten 4 Bits zur Programmierung der Startadresse in 16 verschiedenen 8-KByte-Segmenten haben.
- Alle I/O-Adapter sollten programmierbare I/O-Adressen haben, damit Adreßkonflikte mit anderen I/O-Adaptern auf ein Minimum beschränkt werden können.

Bild 7 zeigt die schematische Vorgehensweise bei der Implementierung des POS-Verfahrens.

Typische POS Implementation

- Disabeln der Interrupts
- Auswählen eines Adapters mit der $\overline{\text{CD Setup(n)}}$ Leitung durch Beschreiben des Setup Registers 96H
- Lesen der Adapter ID von Port 100H und 101H
- I/O Write zu Port 102H mit Bit 0 = 0 um den Adapter zu disabeln und in Setup Mode zu bringen
- POS Daten in Register 103H - 105H schreiben
- I/O Write zu Port 102H mit Bit 0 = 1 zusammen mit den POS Daten für diese Register
- Disable Setup durch Beschreiben des Setup Registers 96H mit 00H
- Enable Interrupts

Bild 7. Eine typische POS-Implementation

Die Arbitration (Buszuteilung) im Multimaster-Betrieb

Wie schon erwähnt, erlaubt die Mikro-Kanal-Architektur mehreren Busmastern einen Zugriff auf den gemeinsamen Bus. Das hierfür gewählte Buszuteilungsverfahren besteht aus einer zentralen Buszuteilungslogik (Central Arbitration Point, CAP), die sich auf der System-Leiterplatte befindet, sowie den lokalen Arbitern, die auf jedem Master im System vorhanden sein müssen (*Bild 8*). Der CAP benutzt die sieben Signale $\overline{\text{PREEMT}}$, ARB/$\overline{\text{GRNT}}$, BURST und ARB0...ARB3 um die Busanforderungen aller Master zu koordinieren.

Ein Arbitrationszyklus wird dadurch eingeleitet, daß einer (oder mehrere) der lokalen Arbiter durch Aktivieren der gemeinsamen (Open Kollektor-) Leitung $\overline{\text{PREEMT}}$ eine Busanforderung signalisieren. Sobald der gegenwärtige Master den Bus freigegeben hat, wird daraufhin

vom CAP ein Arbitration-Zyklus eingeleitet, indem ARB/$\overline{GRNT}$ auf HIGH gelegt wird. Daraufhin legen alle lokalen Arbiter, die sich im Moment um eine Buszuteilung bemühen, die ihnen zugewiesene 4-Bit-Arbitration-ID auf den Bus. Die lokalen Arbiter vergleichen nun bitweise, beginnend mit den MSB ihre eigene Arbitration-ID mit den Arbitration-Leitungen (gemeinsam angelegt von allen Arbitern über Open-Kollektor-Treiber) des Systems (*Bild 9*). Sobald ein Arbiter eine Ungleichheit feststellt, legt er alle niederwertigeren Bits sofort auf High (dies ergibt die sog. Pseudo-Arbitration-ID) und scheidet damit aus dem Arbitration-Zyklus aus. Die Gewinner des jeweiligen Arbitration-Zyklus nehmen weiter an der Arbitration mit ihrer Original-Arbitration-ID teil. Der lokale Arbiter, der als erster eine Übereinstimmung seiner Arbitration-ID mit den Arbitration-Leitungen feststellt, hat die Arbitration gewonnen. Dies ist der Adapter mit der niedrigsten Arbitration-ID und der höchsten Priorität. Die Arbiter, die im laufenden Arbitration-Zyklus ausgeschieden sind, nehmen an den darauffolgenden Arbitration-Zyklen wieder teil. *Bild 10* zeigt, wie solch ein lokaler Arbiter aus UND- und ODER-Gattern aufgebaut werden kann.

Daten-Transfer im Burst-Modus

Prinzipiell wird beim Multimasterbetrieb auf dem Mikro-Kanal-Bus zwischen Single-Cycle- und Burst-Transfers unterschieden. Während beim Single-Cycle-Betrieb im Prinzip vor jedem Buszugriff ein Arbitration-Zyklus stattfinden kann, ist es im Burst-Modus möglich, zusammenhängende Daten blockweise im DMA-Betrieb zu übertragen. Adapter, zum Beispiel Festplattenkon-

	ARB3	ARB2	ARB1	ARB0		ARB ID	ZYKLUS
	0	1	0	1		05H	
	0	1	1	0		06H	1
	1	0	1	0		0AH	
Bus ID	0	0	0	0			
	0	1	1	1	(P)	Gewinner	
	0	1	1	1	(P)	Gewinner	2
	1	1	1	1	(P)	Verlierer	
Bus ID	0	1	1	1			
	0	1	0	1		Gewinner	
	0	1	1	1	(P)	Verlierer	3
	1	1	1	1	(P)	Verlierer	
Bus ID	0	1	0	1			

(P) Pseudo Arbitration ID

Bild 9. Das Arbitration-Schema

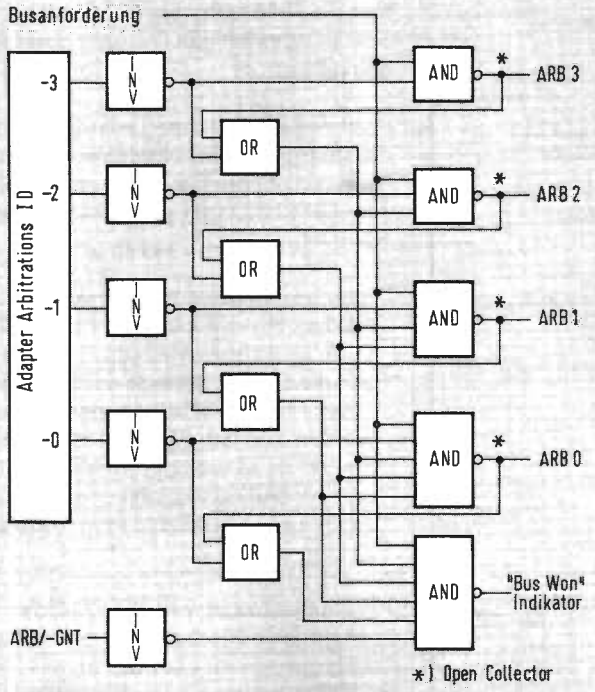
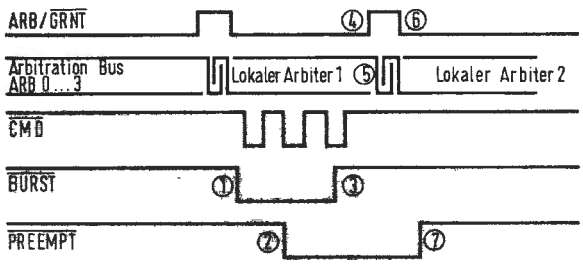


Bild 11. Arbitration im Burst-Modus



① Lokaler Arbiter 1 bekommt Buskontrolle.
② Adapter 2 fordert mit PREEMPT den Bus an.
③ Adapter 1 beendet Burst Transfer und nimmt $\overline{BURST}$ Signal weg.
④ Der "Central Arbitration Point" erkennt Ende des Transfers und legt ARB/$\overline{GRNT}$ auf High
⑤ Die Arbitration beginnt.
⑥ Wenn ARB/$\overline{GRNT}$ auf Low geht, bekommt Lokaler Arbiter 2 die Buskontrolle.
⑦ Adapter 2 nimmt $\overline{PREEMPT}$ als Antwort auf die Buszuteilung weg.

Bild 10. Die Hardware für einen lokalen Arbiter

Arbitration ID	Priorität
-2	Speicher Refresh
-1	NMI
0	DMA Kanal 0*
1	DMA Kanal 1
2	DMA Kanal 2
3	DMA Kanal 3
4	DMA Kanal 4*
5	DMA Kanal 5
6	DMA Kanal 6
7	DMA Kanal 7
8	Reserviert
9	Reserviert
A	Reserviert
B	Reserviert
C	Reserviert
D	Reserviert
E	Reserviert
F	System Microprocessor

Bild 12. Die Prioritätsverteilung für die Arbitration

troller, die Burst-Zyklen auf dem Bus durchführen wollen, müssen über den lokalen Arbiter nach dem Gewinnen der Arbitration die BURST-Leitung aktivieren.

Anschließend können per DMA solange Daten übertragen werden, bis ein anderer Busteilnehmer über die PREEMPT-Leitung den Bus haben will. Nach dem Erkennen eines PREEMPT-Signales darf ein lokaler Arbiter das Burst-Signal noch maximal 7.8 µs aktiv halten. Falls diese Zeit überschritten wird, kann vom System Board ein Timeout ausgelöst werden, indem das ARB/GRNT-Signal auf High gelegt und somit dem gegenwärtigen Busteilnehmer der Bus „entzogen" wird. Anschließend wird ein NMI ausgelöst und Bit 5 und 6 des Arbitration-Registers 90H werden auf High gelegt. Zur Beendigung des Burst-Modus gibt der lokale Arbiter die Burst Leitung nach dem letzten CMD-Puls der Burst-Sequenz wieder frei. *Bild 11* zeigt den prinzipiellen Ablauf einer Bus-Arbitration mit Burst-Übertragung und anschließender Beendigung durch ein PREEMPT-Signal eines anderen Busteilnehmers.

Die Prioritätszuteilung bei der Arbitration

Bild 12 zeigt die Zuteilung der einzelnen Prioritäten. NMI und Refresh, beides interne Signale der System-Leiterplatte, haben die höchste (virtuelle) Priorität -1 und -2. Wenn der CAP einen -1-(NMI)-Request bekommt, aktiviert er PREEMPT, wartet das Ende des laufenden Zyklus ab und legt ARB/GRNT auf High, um alle Adapter vom Bus fernzuhalten. Dieser Zustand wird solange aufrecht erhalten, bis die Ursache der NMI-Auslösung behoben und der NMI selbst zurückgesetzt wurde. Alle DMA-Kanäle, ausgenommen 0 und 4, haben eine feste Arbitration-ID, wobei der DMA-Kanal 0 die höchste und der System-Mikroprozessor die niedrigste Priorität besitzen. Die Arbitration-IDs der Kanäle 0 und 4 sind durch Programmieren des DMA-Kontrollers auf der System-Leiterplatte dynamisch veränderbar. Damit ist es möglich, bei wechselnden Systemanforderungen verschiedenen Adaptern jeweils die höchste DMA-Priorität zuzuteilen.

Bit	Funktion
7	Write/Read
	1 = System Mikroprozessor nimmt an der Arbitration teil.
	0 = System Mikroprozessor nimmt nicht an der Arbitration teil, weil z. B. ein schneller Adapter den gesamten Bus für sich braucht. Diese Bit wird beim Reset auf 0 gesetzt.
6	Write
	1 = Versetzt den CAP in den Arbitrations Status, d.h. der System Mikroprozessor hat die alleinige Kontrolle über den Systembus. Dieses Bit sollte nur von Systemdiagnoseroutinen gesetzt werden.
	0 = Rücksetzen in den normalen Arbitrationsmode.
	Read
	1 = Zeigt an, daß ein NMI aufgetreten ist und über die Arbitration ausmaskiert wurde.
5	Write
	1 = Einstellen erweiterte Arbitration. Damit wird der Arbitrationszyklus von 300ns auf 600 ns verlängert.
	0 = Normale (300 ns) Arbitration. Default Wert.
	Read
	1 = Zeigt Auftreten eines Bus Timeout an. Dieses Bit wird durch Zurückstzen von Bit 6 des Arbitrationsregisters gelöscht, was ein Beenden des Timeout durch Rücksetzen der ARB/-GRNT Leitung bedeutet.
4	Reserviert
3 - 0	Write
	Undefiniert
	Read
	4 Bit Arbitration ID des Busteilnehmers, der gerade den Bus (bekommen) hat. Damit kann der Systemmikroprozessor den Adapters finden, der den Bus Timeout erzeugt hat.

Bild 13. Das zentrale Arbitration-Register

Das POS-Register des zentralen Bus-Arbiters

Bild 13 zeigt die Bedeutung der Bits des zentralen Arbitration-Registers, das über Adresse 90H ansprechbar ist. Die Bits haben unterschiedliche Bedeutung für Lesen und Schreiben.

Das Zeitverhalten des Mikro-Kanals

Das Zeitverhalten der Mikro-Kanal-Architektur basiert auf den Spezifikationen des 10-MHz-80286-Prozessors. Da der 80286 im günstigsten Fall 2 Takte (Ts, Tc) für Speicher- oder I/O-Zyklen benötigt, lag es nahe, dieses Protokoll, soweit es möglich ist, direkt für den Mikro-Kanal zu übernehmen. Für Speicher- oder I/O-Zyklen gibt es daher drei Möglichkeiten:

- Schnellstmöglicher (Default) Zyklus in 2 Takten (200 ns) ohne Wartezyklen. Ein Aktivieren des Verzögerungssignales CH CHRDY ist hier nicht vorgesehen (*Bild 14*) .
- Synchroner erweiterter Zyklus (min. 300 ns). Hierfür kann die CH CHRDY-Leitung aktiv werden, muß jedoch mindestens 30 ns nach der fallenden Flanke von CMD wieder deaktiviert werden, um den Zyklus nicht auf mehr als 300 ns zu verlängern. Modell 50 und Modell 60 verlängern alle Default-Zyklen zu synchronen erweiterten Zyklen (*Bild 15*).
- Asynchroner erweiterter Zyklus (> 300 ns). Hierbei kann der System-Mikroprozessor über das Signal CH CHRDY bis max. 3 µs angehalten werden. Dies ist der allgemeinste Fall für normale Adapterbaugruppen (*Bild 16*).

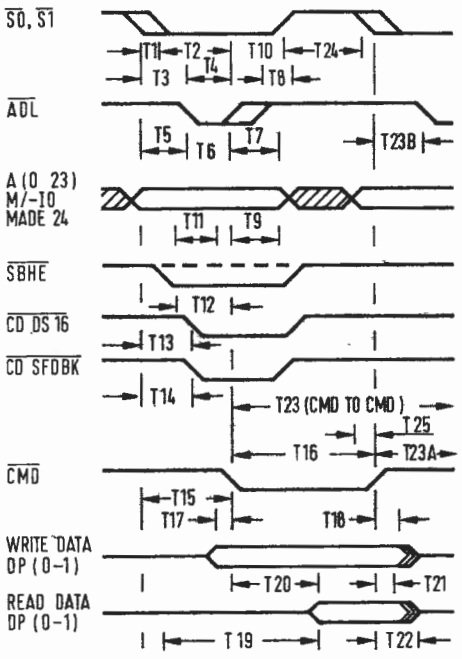
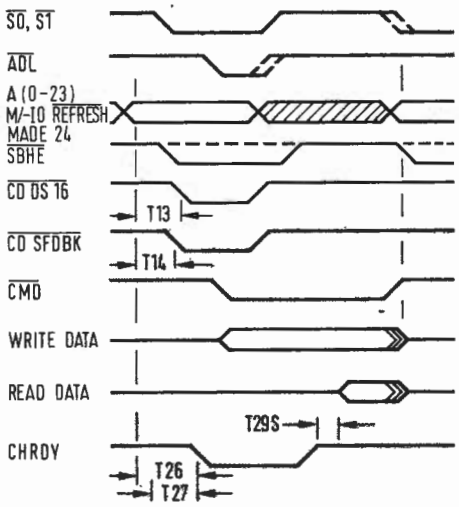


Bild 14. Das Timing zum Default-Zyklus

		Zeiten		Min/Max	
T1	Status active (low)	ab	ADDRESS, M/-IO, -REFRESH valid	10/-	ns
T2	-CND active (low)	ab	Status aktive (low)	55/-	ns 2
T3	-ADL active (low)	ab	ADDRESS, M/-IO, -REFRESH valid	45/-	ns
T4	-ADL active (low)	bis	-CMD active (low)	40/-	ns
T5	-ADL active (low)	ab	Status active (low)	12/-	ns
T6	-ADL pulse width			40/-	ns
T7	Status hold	ab	-ADL inactive (high)	25/-	ns 2
T8	ADDRESS, M/-IO, -REFRESH, -SBHE hold	ab	-ADL inactive	25/-	ns 2
T9	ADDRESS, M/-IO, -REFRESH, -SBHE hold	ab	CMD active (low)	30/-	ns 3
T10	Status hold	ab	-CMD active	30/-	ns 2
T11	-SBHE setup	bis	-ADL inactive	40/-	ns 2
T12	-SBHE setup	bis	-CMD active	40/-	ns 2
T13	-CD DS 16 active (n)	ab	ADDRESS, M/-IO, -REFRESH valid	-/55	ns 3
T14	-CD SFDBK active	ab	ADDRESS, M/-IO, -REFRESH valid	-/60	ns 1
T15	-CMD active	ab	ADDRESS valid	85/-	ns 2
T16	-CMD pulse width			90/-	ns
T17	WRITE DATA setup	bis	-CMD active (low)	0/-	ns
T18	WRITE DATA hold	ab	-CMD inactive (high)	30/-	ns
T19	STATUS	bis	READ DATA valid	-/125	ns
T20	READ DATA valid	ab	-CMD active	-/60	ns
T21	READ DATA hold	ab	-CMD inactive (high)	0/-	ns.
T22	READ DATA BUS tri-state	ab	-CMD inactive (high)	-/40	ns
T23	-CMD active	bis	next -CMD active	190/-	ns
T23 A	-CMD inactive	bis	next -CMD active	80/-	ns
T23 B	-CMD inactive	bis	next -ADL active	40/-	ns
T24	Next STATUS active (low)	ab	STATUS inactive	30/-	ns
T25	Next STATUS active (low)	bis	-CMD inactive	-/20	ns

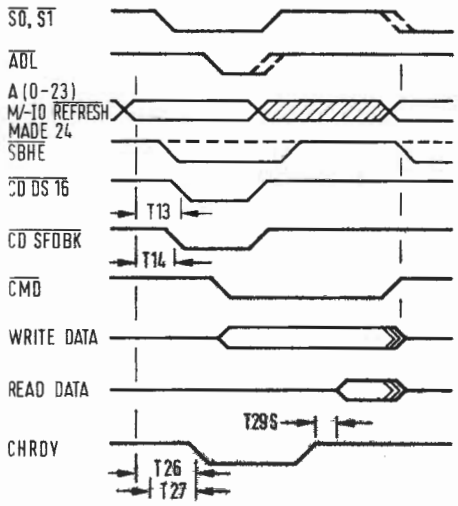
Anmerkungen:
"-" kennzeichnet Signale, die bei 0 aktiv sind.
1.) Alle Slave-Baugruppen müssen -CD SFDBK aktivieren, wenn sie vom System-Mikroprozessor oder dem DMA-Kontroller angesprochen werden. Das gilt nicht für den Setup-Zyklus.
2.) Slave-Baugruppen sollten Informationen mit transparenten Latches mit -ADL oder der fallenden Flanke von -CMD übernehmen.
3.) -CD DS 16 und -CD SFDBK müssen mit ungelatchten Adreßdecodern versehen werden, da die nächste Adresse schon im laufenden Zyklus kommen kann.



		Zeiten		Min/Max	
T13	-CD DS 16 (n) active (low)	ab	ADDRESS, M/-IO, REFRESH valid	-/55	ns
T14	-CD SFDBK (n) active (low)	ab	ADDRESS, M/-IO, REFRESH valid	-/50	ns
T26	CD CHRDY (n) inactive (low)	ab	ADDRESS valid	-/80	ns
T27	CD CHRDY (n) active (low)	ab	Status active	-/30	ns
T29S	Read Data from Slave valid	ab	CD CHRDY n active (high)	-/80	ns

Alle nicht genannten Zeiten sind die aus dem Default-Zyklus

Bild 15. Synchroner, erweiterter Zyklus



		Zeiten		Min/Max	
T13	-CD DS 16 (n) active (low)	ab	ADDRESS, M/-IO, REFRESH valid	-/55	ns
T14	-CD SFDBK (n) active (low)	ab	ADDRESS, M/-IO, REFRESH valid	-/50	ns
T26	CD CHRDY (n) inactive (low)	ab	ADDRESS valid	-/80	ns
T27	CD CHRDY (n) active (low)	ab	Status active	-/30	ns
T29S	Read Data from Slave valid	ab	CD CHRDY n active (high)	-/80	ns

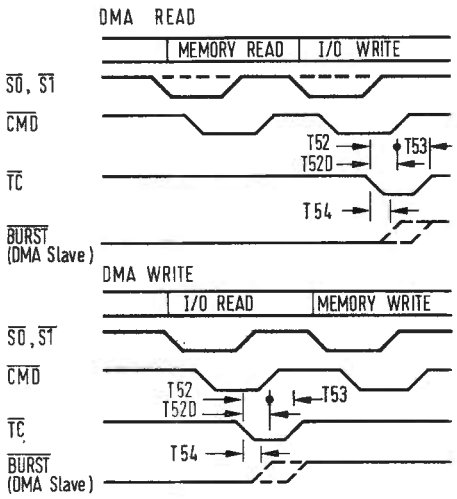
Alle nicht genannten Zeiten sind die aus dem Default-Zyklus

Bild 16. Asynchroner, erweiterter Zyklus

Bild 17 und *18* zeigen das Zeitverhalten im DMA-Betrieb, einmal den Single-DMA-Zyklus ohne Burst, zum Anderen einen Burst-DMA-Zyklus, beide jeweils vom DMA-Controller über $\overline{TC}$ beendet. Hierbei wird der günstigste 200-ns-Zyklus ohne Aktivierung der CH CHRDY-Leitung angenommen.

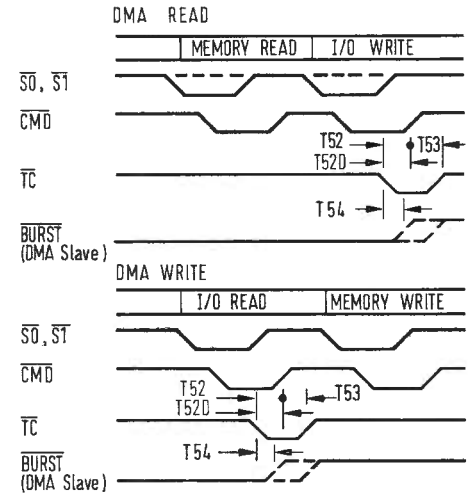
Bild 19 zeigt das Zeitverhalten eines Arbitration-Zyklus. Das Fenster nach Verlassen des PREEMPT-Status ist dazu da, anderen Adaptern die Möglichkeit zu geben, das inaktive PREEMPT zu erkennen und den inaktiven Status zu verlassen.

Zusammenfassend kann gesagt werden, daß der Mikro-Kanal-Bus aufgrund seiner Geschwindigkeit, seiner Multimasterfähigkeit und seiner Konfigurationsmöglichkeiten ein ernst zu nehmender Konkurrent für alle Multimaster-Busse (wie VME, Multibus I/II oder AMS) werden wird. Ob alle Anwender von den bisherigen Systemen mit dem nicht multimasterfähigen PC/IO Bus zu der neuen Architektur wechseln werden, bleibt abzuwarten.



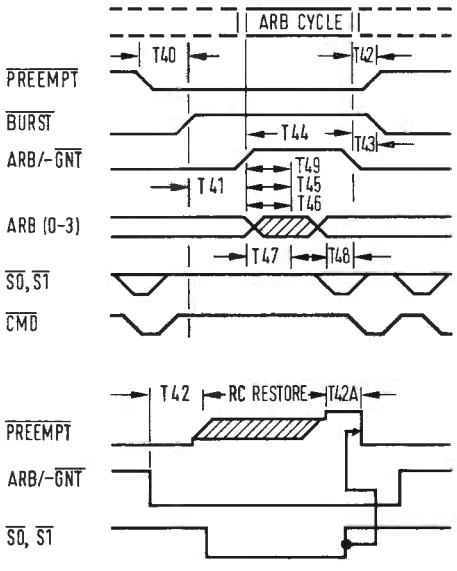
		Zeiten		Min/Max
T 52	-TC setup	bis	-CMD inactive	30/– ns
T 52D	-TC setup	bis	-CMD inactive	15/– ns
T 53	-TC hold	ab	-CMD inactive	10/– ns
T 54	-BURST released by DMA slave	ab	-TC active	–/30 ns

Bild 17. Single-DMA-Transfer (vom DMA-Kontroller beendet)



		Zeiten		Min/Max
T 52	-TC setup	bis	-CMD inactive	30/– ns
T 52D	-TC setup	bis	-CMD inactive	15/– ns
T 53	-TC hold	ab	-CMD inactive	10/– ns
T 54	-BURST released by DMA slave	ab	-TC active	–/30 ns

Bild 18. Burst-DMA-Transfer (vom DMA-Konbtroller beendet)



		Zeiten		Min/Max
T40	-PREEMPT active (low)	bis	End of Transfer	0/7,8 µs
T41	ARB/-GNT high	ab	End of Transfer	30/– ns
T42	-PREEMPT inactive (high)	ab	ARB/-GNT low	0/50 ns
T42A	-PREEMPT inactive (high)	bis	Status inactive (Exiting Inactive Status)	20/– ns
T43	-BURST active (low)	ab	ARB/-GNT low (By Bursting DMA Slave)	–/50 ns
T44	ARB/-GNT high			300/– ns
T45	Driver turn on delay	ab	ARB/-GNT high	0/50 ns
T45A	Driver turn on delay	ab	lower priority line	0/50 ns
T46	Driver turn off delay	ab	ARB/-GNT high	0/50 ns
T47	Driver turn off delay	ab	higher priority line	0/50 ns
T48	Arbitration bus stable	bis	ARB/-GNT low	10/– ns
T49	Tri-state drivers	ab	ARB/-GNT high	–/50 ns

Bild 19. Arbitration-Zeitverhalten

Ingrid Trommer

Einführung in Unix

Teil 3: Benutzeroberfläche Shell

Die Shell interpretiert die Eingaben des Anwenders und bringt diese zur Ausführung. Kurz ausgedrückt, ist die Shell das, was der Anwender von Unix sieht. In diesem Teil unserer Serie erklärt unsere Autorin wichtige Details der Bourne-Shell.

Außer der Bourne-Shell, benannt nach ihrem Entwickler S. R. Bourne, gibt es bei vielen Systemen auch die C-Shell, die in Berkeley entwickelt wurde. Der Name entstand aus der Tatsache heraus, daß ihre Syntax der Sprache C sehr ähnlich ist. Da die Bourne-Shell weiter verbreitet ist, soll auf die C-Shell hier nicht näher eingegangen werden.

Standard-Ein-/Ausgabe und Hintergrundverarbeitung

Die Shell ist ein Kommandozeilen-Interpreter, d. h. jeder Befehl muß erst durch <Return> abgeschlossen werden, bevor er abgearbeitet wird. Die Shell liest eine Zeile von der Tastatur, wandelt eventuelle Metazeichen (siehe unten) um und ruft das entsprechende Kommando mit Parametern auf. Die Ausgabe erfolgt auf dem Monitor. Die einzelnen Befehle erwarten eventuelle Eingaben von der Tastatur, Ausgaben erscheinen auf dem Monitor. In Unix gibt es dafür die Begriffe Standard-Eingabe und Standard-Ausgabe. Das System stellt noch einen weiteren Kanal zur Verfügung: Den Fehlerausgang, auch Standard-Fehler oder neudeutsch Standard-Error genannt, der auch mit dem Monitor verbunden ist.

Diese Zusammenhänge werden von *Bild 1* verdeutlicht.
Gibt der Benutzer einen Befehl über die Tastatur ein und schließt ihn mit <Return> ab, so erscheint der nächste Prompt erst dann, wenn der Befehl abgearbeitet ist. Dauert das Abarbeiten des Befehls länger, so ist es lästig, daß man in dieser Zeit das Terminal nicht zur Verfügung hat. Hier kennt Unix eine Abhilfe! Der Befehl wird im Hintergrund (Background) gestartet, und der Benutzer hat das Terminal sofort wieder zum Weiterarbeiten zur Verfügung. Die Syntax für Hintergrund-Verarbeitung ist „&" nach dem normalen Befehl. Das System gibt dann eine Zahl zurück, die sogenannte Prozeßidentifikation. Mit dieser Prozeßidentifikation kann der Anwender den Prozeß evtl. vorzeitig beenden. Es soll schon vorgekommen sein, daß man erst nach erfolgter Eingabe merkte, daß die Eingabe unsinnig war und der Rechner ganz fürchterliche Dinge anstellt. Das Kommando „kill", das als Parameter die Prozeßidentifikation benötigt, dient dazu, einen Befehl vorzeitig zu beenden. *Bild 2* zeigt ein Beispiel für die Eingabe eines Befehls, der im Hintergrund abgearbeitet werden soll, und das anschließende Stoppen dieses Befehls.

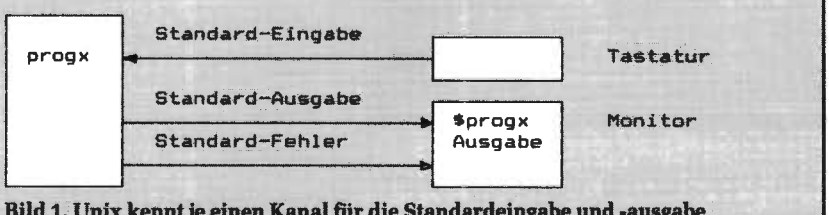


Bild 1. Unix kennt je einen Kanal für die Standardeingabe und -ausgabe und den Standardfehler

```
$ cat *.c > /dev/lp &
15346
$ ps
   PID TTY TIME COMMAND
 15347  co 0:01 ps
 15346  co 0:00 cat
 15135  co 0:06 sh
$ kill 15346
$ ps
15346 Terminated
   PID TTY  TIME COMMAND
 15349  co  0:01 ps
 15135  co  0:06 sh
$
```

Bild 2. So wird der „kill"-Befehl angewandt

Das Kommando „ps" (process status) zeigt, welche Prozesse des Anwenders momentan abgearbeitet werden, sowie deren Prozeßidentifikation.

Umlenkung der Ein-/Ausgabe

Da die Shell die Kommandozeile interpretiert, bevor sie dem Rechner zur Bearbeitung geschickt wird, ist es möglich, die drei Standard-Kanäle auf andere Geräte oder auf Dateien zu legen. Diese Fähigkeit der Umlenkung (Redirection) ist eines der Dinge, welche die Shell zu einem solch mächtigen Werkzeug macht.
Wie sieht die Syntax aus?
Input-Redirection: „progx < dat.ein"
Output-Redirection: „progx > dat.aus"
Auch die Verknüpfung beider ist möglich:
„progx < dat.ein > dat.aus"
Da Dateien unter Unix von der Syntax her genauso wie Geräte behandelt werden, kann die Umleitung auch von oder auf Geräte erfolgen. So wird z. B. mit „progx >/dev/lp" die Standardausgabe auf den Drucker umgelenkt. Das Symbol für Standardfehler ist 2. Die Umlenkung vom Standardfehler wird wie folgt angegeben:
„progx 2 > /dev/lp" (siehe auch *Bild 3*)
Erfolgt die Umleitung der Standardausgabe auf eine Datei, so erstellt die Shell diese Datei neu, falls sie nicht bereits vorhanden ist. Existiert diese Datei, so wird sie überschrieben. Selbstverständlich ist es möglich, das Überschreiben zu verhindern. Die Syntax dafür ist „>>". Mit „progx >> dat.aus" wird die Ausgabe von progx an die Datei dat.aus angehängt. *Bild 4* zeigt das Zusammenspiel von Ein- und Ausgabeumlenkung. Der Befehl „sort" sortiert eine Datei und gibt das Ergebnis auf der Standardausgabe aus. Die Option „-r" sortiert statt in aufsteigender in absteigender Reihenfolge. Das Beispiel zeigt die Ausgabe der Datei „zahlen" mit „cat". Der „sort"-Befehl

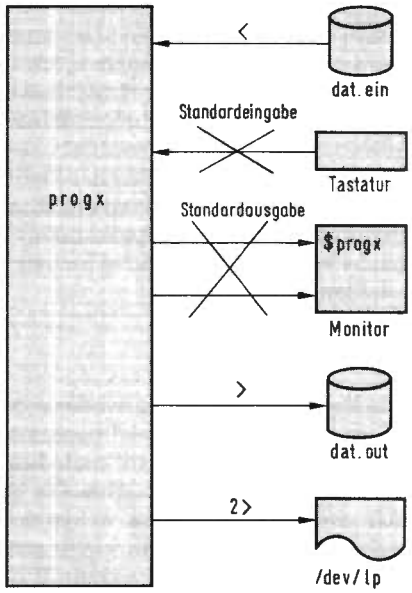


Bild 3. Ein-/Ausgaben lassen sich leicht umlenken

```
$ cat zahlen
1
2
3
4
5
$ sort -r < zahlen > zahlenr
$ cat zahlenr
5
4
3
2
1
$
```

Bild 4. Umlenkung von Ein- und Ausgabe

liest, statt von der Standardeingabe, von der Datei „zahlen" und gibt die Elemente der Datei, statt auf der Standardausgabe, auf die Datei „zahlenr" aus.

Filter und Pipes

Unix-Kommandos sind meist gleichartig aufgebaut. Sie lesen von der Standardeingabe, bearbeiten diese Eingabe und schreiben ihre Ausgabe auf die Standardausgabe. Ein so aufgebautes Kommando nennt man Filter. Man könnte sich nun vorstellen, Filter über Temporärdateien miteinander zu verbinden. Also Programm X schreibt auf eine Temporärdatei, Programm Y liest von dieser Temporärdatei, verarbeitet also die Ausgabe von Programm X weiter.
Unix nimmt dem Benutzer das Anlegen der Temporärdatei ab. Die Syntax ist, wie gewohnt, sehr kompakt.
„progx|progy" bedeutet somit, daß progy die Standardausgabe von progx als Standardeingabe benutzt. Das nennt man unter Unix eine Pipe. Selbstverständlich können über eine Pipe nicht nur zwei Programme miteinander verbunden werden. Die Anzahl der miteinander verbundenen Programme ist beliebig, solange sie die Unix-Konvention einhalten; nämlich: Lesen von Standard-Eingabe und schreiben auf Standard-Ausgabe. Wenn Programme durch Pipes miteinander verbunden werden sollen, müssen sie Filter sein! Es können dabei durchaus vom System zur Verfügung gestellte Tools, mit selbst geschriebenen Programmen durch Pipes verbunden werden.
Bild 5 veranschaulicht den Zusammenhang.
Bild 6 zeigt die Anwendung einer Pipe an einem konkreten Beispiel. Das Programm „wc" für „word count" zählt die Buchstaben, Worte und Zeilen einer Datei. Mit der Option „-l" werden nur die Zeilen gezählt. Zuerst arbeitet „wc" mit der Datei „zahlen". Die Datei enthält fünf Zeilen. Der Befehl „who" zeigt, welcher Benutzer an welchem Terminal arbeitet. Für jeden Benutzer entsteht eine Zeile. Wenn nur die Anzahl der momentanen Benutzer interessiert, so kann man die Ausgabe von „who" als Eingabe für „wc -l" benutzen. Damit erhält man die Anzahl der momentanen Benutzer.

Meta-Zeichen

Wird am Terminal z. B. ein „x" eingegeben, so wird dieses Zeichen von der Shell auch als „x" interpretiert. Es gibt aber Zeichen, die eine besondere Funktion haben, also eine zusätzliche Bedeutung zu ihrem eigentlichen Zeichenwert. Diese Zeichen nennt man Meta-Zeichen. Das gebräuchlichste dieser Zeichen ist der Asterisk (*), ein weiteres das Fragezeichen. Meta-Zeichen werden von der Shell benutzt, um Dateinamen zu generieren. Bei anderen Betriebssystemen nennt man den Asterisk auch „wild card symbol". Der Asterisk ist für sich allein ein Synonym, das für alle Namen in einem Dateiverzeichnis steht. Das Fragezeichen steht für ein einzelnes Zeichen. Man kann die Meta-Zeichen auch mit „normalen" Buchstaben oder weiteren Meta-Zeichen kombinieren, um bestimmte Dateinamen zu generieren.

```
$ wc -l zahlen
       5 zahlen
$ who
hugo         console  Feb 11 18:02
root         tty01    Feb 11 18:26
$ who | wc -l
       2
$
```

Bild 6. Ein einfaches Pipe-Beispiel

Einige Beispiele sollen dies verdeutlichen. Wir gehen davon aus, daß das momentane Directory die in *Bild 7* gezeigten Dateien enthält.
Will man die Namen aller Dateien sehen, die mit „ar" anfangen, so lautet die Eingabe, wie aus *Bild 8* zu ersehen, „ls ar*". Aus dem Beispiel in *Bild 9* geht hervor, daß der Asterisk auch für kein Zeichen steht. Fragt man nach „absatz*", so wird auch der Name „absatz" selbst mitausgegeben.
In *Bild 10* wird die Kombination von Meta-Zeichen und „normalen" Zeichen gezeigt.
Das Fragezeichen steht im Gegensatz zum Asterisk für genau ein Zeichen (*Bild 11*) Mit „ab*.??" gibt der „ls"-Befehl alle Dateinamen aus, die mit „ab" anfangen, dann nach beliebigen Zeichen den Punkt enthalten und nach dem Punkt genau zwei weitere Zeichen.
Neben diesen Metazeichen, die auch MS-DOS-Anwendern geläufig sind, kennt Unix noch eine weitere Möglichkeit, Dateinamen zu generieren. Für ein Zeichen an einer bestimmten Stelle kann in eckigen Klammern ein Gültigkeitsbereich angegeben werden. Mit „[a-c]" sind an dieser Stelle entweder „a", oder „b", oder „c" erlaubt. Mit Zahlen kann genauso verfahren werden

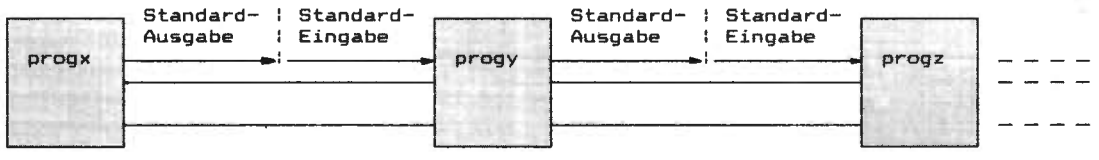


Bild 5. Auch MS-DOS-Anwendern vertraut: die Pipe

(siehe *Bild 12*). Das zweite Beispiel in *Bild 13* zeigt, daß ein Bereich wirklich nur für ein Zeichen gültig ist. Der Name „absatz2.9" wird nicht erkannt.
Das Erzeugen von Dateinamen ist nicht nur für den Befehl „ls" nützlich. In vielen Fällen helfen die Metazeichen unnötige Schreibarbeit zu vermeiden. Man denke z. B. an den Befehl „rm". Mit „rm *" kann man alle Dateien im momentanen Dateiverzeichnis löschen. Wie bereits erwähnt, führt die Shell diesen Befehl ohne unnötige Rückfragen, wie etwa „Are you sure", aus und meldet sich mit dem nächsten Prompt. Mit „rm ar*" löscht man alle Dateien, die mit „ar" anfangen, und, falls vorhanden, auch die Datei „ar". Was passiert, wenn man sich vertippt? Also statt „rm ar*", „rm ar *".
Jetzt wird es tragisch. Zuerst kommt die Fehlermeldung, daß die Datei „ar" nicht gelöscht werden kann, weil sie nicht existiert. Anschließend werden alle Dateien des momentanen Directories gelöscht!

```
$ ls absatz2.[1-10]
absatz2.1
$
```

Bild 13. Ein Bereich steht nur für ein einzelnes Zeichen

Durch die irrtümliche Eingabe des Leerzeichens generiert die Shell aus dem alleinstehenden Asterisk alle Dateinamen des momentanen Directories. Hier kann man sich allerdings vor sich selbst schützen. Benutzt man die „-i" Option des Befehls „rm" („-i" steht für interaktiv), so wird jedesmal der Dateiname mit einem Fragezeichen ausgegeben. Gelöscht wird erst, wenn man mit „y" für yes geantwortet hat. Alle anderen Eingaben bewirken, daß die Datei erhalten bleibt.

Shell-Variable

Alles was bis hierher über die Shell gesagt wurde, unterscheidet sich nicht sehr von herkömmlichen Kommando-Interpretern. Die beiden folgenden Abschnitte zeigen, wo sich die Shell unterscheidet und was sie zu einem solch mächtigen Werkzeug macht. Mit diesen Werkzeugen ist es möglich, Variable zu bilden oder auf System-Variable zuzugreifen. Der eigentliche Wert der Shell wird erst offenbar, wenn man die Möglichkeiten der Shell als Programmiersprache kennt. Doch zunächst zu den Variablen.

System-Variablen

Diese Variablen werden entweder vom System oder vom System-Administrator belegt. Teilweise können sie auch durch den Benutzer verändert werden. Die Variable „HOME" enthält den Namen des Dateiverzeichnisses, in dem ein Benutzer anfängt, nachdem er die Login-Prozedur erfolgreich durchlaufen hat. Das Home-Directory wird durch den System-Administrator zugewiesen.
Die Variable „PATH" besteht aus Namen von Dateiverzeichnissen, die durch „:" voneinander getrennt sind. Gibt der Benutzer einen Befehl, also z. B. „pwd", so werden alle Directories im PATH der Reihe nach durchsucht, ob ein ausführbares Programm mit dem Namen „pwd" enthalten ist. Wird „pwd" gefunden, so wird es ausgeführt. Normalerweise hat die Variable PATH eine Vorbelegung, sie kann aber vom Benutzer verändert werden. Will man dem Standardpfad noch ein Directory hinzufügen, das unter dem Home-Directory liegt und „exec" heißt, so geschieht das mit der folgenden Syntax (die Eingaben des Anwenders sind mit dem Unterstrich gekennzeichnet):

```
$ PATH=$PATH:$HOME/exec
$
```

Daraus ist ersichtlich, wie zwischen der Variablen selbst (man könnte auch sagen ihrer Adresse) und ihrem Wert unterschieden wird. Den Wert einer Variablen erhält man durch das vorangestellte Dollar-Zeichen, die Zuweisung erfolgt über die Adresse, also dem Namen der Variablen. In einer Wertzuweisung dürfen keine Leerzeichen enthalten sein.
Oft ist das momentane Dateiverzeichnis nicht im Pfad enthalten. Das führt bei ungeübten Benutzern oft zu Verwirrung. Mit „ls -l" kann man sehen, daß es ein ausführbares Programm mit dem angegebenen Namen im momentanen Directory gibt. Das System behauptet dennoch, dieses Programm nicht finden zu können. Nur ausführbare Programme, deren Dateiverzeichnisse im „PATH" enthal-

```
$ ls
absatz      absatz2.1   absatz3.11   absatz3.2   artikel2
absatz1.1   absatz2.10  absatz3.12   absatz3.5   artikel13
absatz1.2   absatz2.9   absatz3.17   absatz4.1   artikel14
absatz1.3   absatz3.1   absatz3.18   artikel11
$
```

Bild 7. Ein Dateiverzeichnis

```
$ ls ar*
artikel1  artikel2  artikel3  artikel4
$
```

Bild 8. Das Metazeichen Asterisk reduziert lange Eingaben

```
$ ls absatz*
absatz      absatz1.3    absatz2.9    absatz3.12   absatz3.2
absatz1.1   absatz2.1    absatz3.1    absatz3.17   absatz3.5
absatz1.2   absatz2.10   absatz3.11   absatz3.18   absatz4.1
$
```

Bild 9. „absatz*" steht auch für „absatz" allein

```
$ ls a*.*
absatz1.1   absatz2.1   absatz3.1    absatz3.17   artikel3.5
absatz1.2   absatz2.10  absatz3.11   absatz3.18   artikel4.1
absatz1.3   absatz2.9   absatz3.12   artikel3.2
$
```

Bild 10. Kombination von Metazeichen und „normalen" Zeichen

```
$ ls ab*.??
absatz2.10  absatz3.11  absatz3.12  absatz3.17  absatz3.18
$
```

Bild 11. Ein „?" steht für jeweils ein einzelnes Zeichen

```
$ ls absatz3.[1-2][1-9]
absatz3.11  absatz3.12  absatz3.17  absatz3.18
$
```

Bild 12. Die Anwendung von Bereichen (hier mit Zahlen)

ten sind, werden vom System gefunden und ausgeführt!
Wie so oft, gibt es auch hier eine Ausnahme. Ein ausführbares Programm kann absolut adressiert werden und sich damit aufrufen lassen. Mit der Eingabe von „/bin/date" umgeht man „PATH" und erreicht die Ausführung.
Soll grundsätzlich das momentane Directory im Suchpfad enthalten sein, so sieht die Syntax dafür so aus:

```
$ PATH=$PATH:.
$
```

Der Punkt am Ende der Eingabe steht für das momentane Directory. Die Systemvariablen „PS1" und „PS2" stehen für den primären und den sekundären Prompt. Der sekundäre Prompt erscheint dann, wenn die Shell der Meinung ist, daß ein Befehl nicht bis zum Ende eingegeben wurde. Beide Prompts können durch den Benutzer verändert werden. Hier ein Beispiel:

```
$ PS1=hallo
hallo
```

Variable sind grundsätzlich nur in der momentanen Shell wirksam. Ruft der Anwender aus der Shell eine Prozedur auf, so wird diese in einer Sub-Shell ausgeführt. Variable der aufrufenden Shell sind also nicht bekannt. Es gibt aber einen Mechanismus, um Variable nach unten durchzureichen.

Der Befehl dazu heißt „export". Mit

```
$ export PATH
$
```

wird der Suchpfad in alle Unter-Shells exportiert. Mit der Eingabe „set" kann man die System-Variablen abfragen. Die Ausgabe von „set" ist, je nach Hersteller, unterschiedlich. *Bild 14* zeigt ein Beispiel.
Falls nur der Wert einer Variablen ausgeben werden soll, so kann man das mit dem Befehl „echo" erreichen. „echo" gibt nachfolgende Worte aus.

```
$ echo hallo
hallo
$ echo $PATH
/bin:/usr/bin:/etc:.
$
```

Vorbelegte Variable

Von den vorbelegten Variablen sollen hier nur die wichtigsten aufgelistet werden. Wie sie angewendet werden können, zeigt der Absatz über die Shell-Programmierung in der nächsten Folge.

```
$ set
HOME=/usr/hugo
LOGNAME=hugo
MAIL=/usr/mail/hugo
PATH=/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin:
PS1=$
PS2=>
SHELL=/bin/sh
TERM=hp2392a
TZ=GMT0
$
```

Bild 14. Mit „set" werden die Werte der System-Variablen abgefragt

$? Der Rückgabewert des letzten Befehls.
$$ Die Prozeßnummer des momentanen Prozesses.
$! Die Prozeßnummer des letzten im Hintergrund gestarteten Befehls.
$0 Der Name mit dem der momentane Befehl aufgerufen wurde.
$1 $2 ... $9 Die positionalen Parameter. Parameter die beim Aufruf eines Befehls übergeben werden können.
$# Die Anzahl der übergebenen positionalen Parameter.

Eigendefinierte Variable

Man kann sich auch selbst Variablen definieren. Die Definition einer Variablen erfolgt durch eine Wertzuweisung:

```
$ var=17
$ echo $var
17
$
```

Eine sinnvolle Anwendung ergibt sich z. B., wenn der Anwender eine Zeitlang immer wieder mit dem Namen eines sehr langen Dateiverzeichnisses arbeiten muß. Wenn er diesen Namen in eine Variable schreibt, kann er sich viel Schreibarbeit sparen.
Mit den folgenden Befehlen wird die Datei „datei1" unter dem Verzeichnis „/usr/hugo/oha/arg/langer/name" in das momentane Directory kopiert:

```
$ dir=/usr/hugo/oha/arg/langer/name
$ cp $dir/datei1 .
$
```

Assembler-Inlines in C unter CP/M-68k

Gelegentlich besteht der Bedarf, Assembler-Befehle in C-Programme einzufügen. Viele Compiler besitzen dafür die Anweisung #asm und #endasm, die Anfang und Ende von Assembler-Teilen in C-Programmen markieren. Beim C-Compiler von Digital Research gibt es dafür die nicht dokumentierte asm-Funktion. Beispielsweise kann man mit der Anweisung asm("jsr $ff7424") beim mc-68000-Computer die Statuszeile ausschalten. Das Problem ist jetzt nur noch die Übertragung von C-Variablen in die Register des Prozessors 68000. Dies kann man mit C-Variablen der Speicherklasse Register leicht erreichen: die erste Register-Variable eines C-Programms wird im Register D7 (= R7) angelegt, die zweite im Register D6 (= R6) usw. Das Bild zeigt ein Anwendungsbeispiel.

Dr. Johannes-Heinrich Vogeler

```
char *text;
register long xx;
text = (char *)malloc(81);
strcpy(text,"So wird in die Statuszeile geschrieben!");
xx = text;
asm("clr.w d1");
asm("move.l d7,a0");  /* a0 := Uebergaberegister fuer String */
asm("jsr $ff7410");
```

Schreiben in die Statuszeile des mc-68000-Computers: Mit asm-Funktion und Registervariablen ist das auch in C-Programmen kein Problem

Guido Dampf

Transputer

Ein Konzept aus Europa

Was Transputer sind und was sie leisten und vor allen Dingen, welches neue (?) Konzept dahinter steckt, soll in diesem Beitrag ausführlich beschrieben werden.

Man nehme : Eine schnelle 32-Bit-CPU mit eingebautem Dispatcher, einige Timer, einen 64-Bit-Arithmetikprozessor, 4 KByte schnelles RAM, vier, bis zu 2,3 MByte/s schnelle serielle, DMA-fähige Schnittstellen sowie eine ausgefeilte Bus- und Peripherielogik zum Anschluß von bis zu 4 GByte externen Speicher und DMA-fähiger Peripheriegeräte, setze all das auf einen einzigen Chip und ... ? (*Siehe Bild 1*)

Das alles ist jedenfalls auf dem neuen Transputerchip IMST800 von INMOS in 1,5-micron-CMOS-VLSI-Technologie integriert. Dabei ist dieser Chip auch intern noch sehr, sehr schnell. Additionen in 64-Bit-real-Arithmetik benötigen knappe 233 ns, Multiplikationen 433 ns. Beim Whetstone-Benchmarktest erreicht

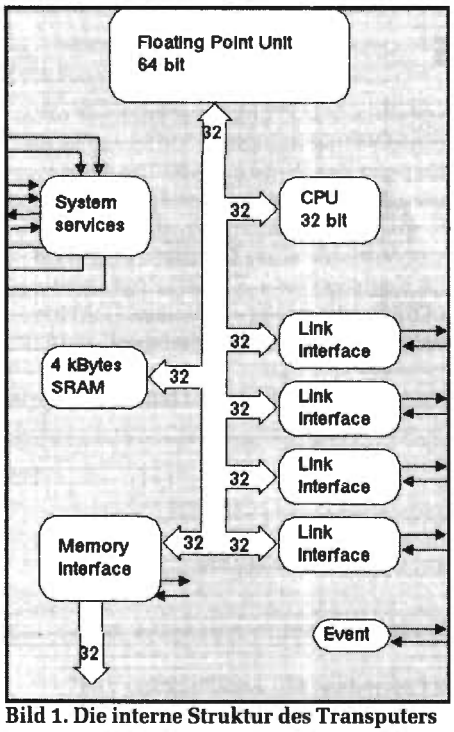


Bild 1. Die interne Struktur des Transputers TMS800

der IMST800-30 den 20fachen Durchsatz eines IBM-ATs mit 80287-Coprozessor ! Selbst der kleine Bruder des T800, der T414 schafft den „Sieve-Benchmark" (Ganzzahl-Arithmetik, Sieb des Erathostenes) bis zu 56 mal schneller als ein IBM-PC. Das sind schwindelerregende Zahlen, doch was ist an Transputern ein neues Konzept?

transmit und compute

Das Wort „Transputer" übersetzt man wohl am besten mit „übermittle und rechne" (transmit & compute), denn wirklich neu ist die Aufwertung der Datenübermittlung als integraler Bestandteil der Hardware-Architektur. Anders als bei herkömmlichen Computer-Netzwerken, deren Hauptaufgabe es ist, allen angeschlossenen Rechnern den Zugriff auf gemeinschaftlich nutzbare Daten zu gewährleisten, sowie gerade nicht genutzte Rechner für rechenintensive Hintergrundarbeit zu benutzen, laufen die Prozesse in einem Transputersystem zwar physikalisch auf mehrere Chips verteilt, aber eben nicht so autonom wie in einem Netzwerk ab. Die Prozesse stehen vielmehr ständig miteinander in Verbindung und übermitteln sich gegenseitig die berechneten Ergebnisse. Dabei arbeiten alle Transputer gemeinschaftlich an einem Problem. Dieses wird dafür so zerlegt, daß die entstehenden Teilaufgaben parallel abgearbeitet werden können. So soll das Gesamtproblem in möglichst kurzer Zeit bearbeitet werden. Jede Teilaufgabe wird von einem eigenen Prozeß bearbeitet, der im Idealfall logisch und physikalisch von den anderen Prozessen getrennt ist. Nur über sogenannte Links gibt es Querverbindungen. Ein Prozeß arbeitet mit eigenem Speicher und eigener CPU. Nur Daten und Signale werden mit anderen Prozessen ausgetauscht. In einem Netzwerk dagegen muß ein ausgelagertes Problem ein vollständiges Programm sein, welches erst wenn es beendet ist sein Ergebnis zurückmeldet. „Teamwork" der Transputer bei der Lösung eines Problems macht erst den wahren Geschwindigkeitsvorteil aus. Es ist nicht mehr die Taktgeschwindigkeit der Prozessoren, die eine Geschwindigkeitsbegrenzung bestimmt, sondern die Frage, in wieviele parallel ablauffähige Teilaufgaben sich das zu lösende Problem zerlegen läßt. Damit dieses Team von Prozessen sauber zusammen arbeiten kann, muß es eine klar definierte Schnittstelle geben, die schon erwähnten Links.

Was Links sind

Ein Link ist eine serielle Schnittstelle mit festem Protokoll, die unabhängig von der Aktivität des CPU-Teils Daten mit einer hohen Geschwindigkeit per DMA (Direkten Speicherzugriff) übermitteln kann. Beim T800 sind das brutto 20 MBit/s = 1,7 MByte/s unidirektional. Jeder Transputer hat vier solche Links integriert, die auch untereinander vollkommen unabhängig sind und einen sehr schnellen Datenaustausch zwischen den Transputern ermöglichen. Da das Hauptaugenmerk beim Transputerkonzept also auf der Interaktion der Prozesse liegt, wurde in die Entwicklung der Transputerlinks auch wesentlich mehr Know-how gesteckt, als es sonst für eine Schnittstelle üblich ist. Dabei ist die Benutzung der Links für den Anwender vollkommen transparent und einfach. Über diese Links steht der Transputer nicht nur mit anderen Transputern in Verbindung, sie können auch als serielle Schnittstelle zur Außenwelt fungieren. So muß zum Beispiel ja irgendwie einmal ein Programm in den Speicher des Transputers geladen werden. Wenn dieses nicht in einem Festwertspeicher (ROM) steht und kein DMA-Zugriff auf den Transputerspeicher möglich ist, muß es eben über ein Link dorthin gelangen. Ferner können darüber zum Beispiel Meßwerte oder Steuerkommandos für numerisch gesteuerte Maschinen übertragen werden. Um die Leistung eines Transputersystems möglichst flexibel zu halten, gaben die Entwickler dem Transputer noch einen besonderen Leckerbissen mit – einen Dispatcher (Prozeßumschalter). Ganz in Hardware aufgebaut, bewirkt der Dispatcher, daß schon auf einem Transputerchip allein mehrere Prozesse quasiparallel ablaufen können. Da die Programmierung der quasiparallelen Prozesse genauso erfolgt wie die echt parallel laufender (also auf verschiedenen Chips

laufender) Prozesse, sind bei einer Erweiterung des Systems um zusätzliche Transputerchips kaum Softwareänderungen durchzuführen (bei einigen Hochsprachensystemen sogar gar keine). Es ist also möglich, auf einem kleinen Entwicklungssystem (wie z. B. einer IBM-PC-Steckkarte) mit ein oder zwei Transputern Software zu schreiben und diese im Ernstfall auf einem System laufen zu lassen, das genau so viele Transputer enthält, wie nötig sind, um die gewünschte Geschwindigkeit zu erreichen.

Multitasking ist eingebaut

Um diese Universalität zu besitzen, muß ein Transputer also von Hause aus „Multitasking" können – und eben dafür ist der Dispatcher zuständig. Er schaltet, abhängig von Timer- sowie internen und externen Zuständen, zwischen den einzelnen Prozessen innerhalb einer Zeit von unter 1 µs um, so daß die CPU fast nie „arbeitslos" wird oder (genauso schlimm) nur mit simplem Polling beschäftigt ist. Der gerade aktive Prozeß wird so lange bearbeitet, bis sich ein Zustand einstellt, der es einem höher priorisierten Prozeß erlaubt, weiterzuarbeiten oder der dem gerade aktiven Prozeß das Weiterarbeiten unmöglich macht (z. B. Warten auf Eingabe/Ausgabe). Spätestens wird so ein Zustand erreicht, wenn der Timer für diesen Prozeß abgelaufen ist.
Um schnell zwischen Prozessen umschalten zu können, verwaltet der Dispatcher drei Register, den Workspace-Zeiger, der auf den gerade zu bearbeitenden Prozeß zeigt, sowie einen Front- und einen Back-Zeiger, die auf den Anfang und das Ende der Liste der wartenden Prozesse zeigen. Die wartenden Prozesse sind über einen in ihre Arbeitsspeicherbereiche eingetragenen Zeiger zu einer Liste verbunden (*Bild 2*). Um die Kompatibilität von quasiparallelen mit parallelen Prozessen sicherzustellen, werden außerdem auch noch Soft-Links verwaltet. Sie sehen vom Programm her genau so aus wie echte „Drähte", sind aber nichts weiter als Speicherplätze, über die die quasiparallelen Prozesse miteinander kommunizieren. Insgesamt betrachtet sind die Transputerchips von INMOS schon sehr mächtige vollständige Computer. Sie haben als solche auch noch einen Bus zum Anschluß von bis zu 4 GByte externen Speicher. Zwischen dem externen Speicher und den internen 4 KByte SRAM besteht vom Programmieren her kein Unerschied, jedoch läuft ein Programm in dem 50 ns-Internspeicher ohne Wartezyklen und somit wohl wesentlich schneller als im externen Speicher ab.

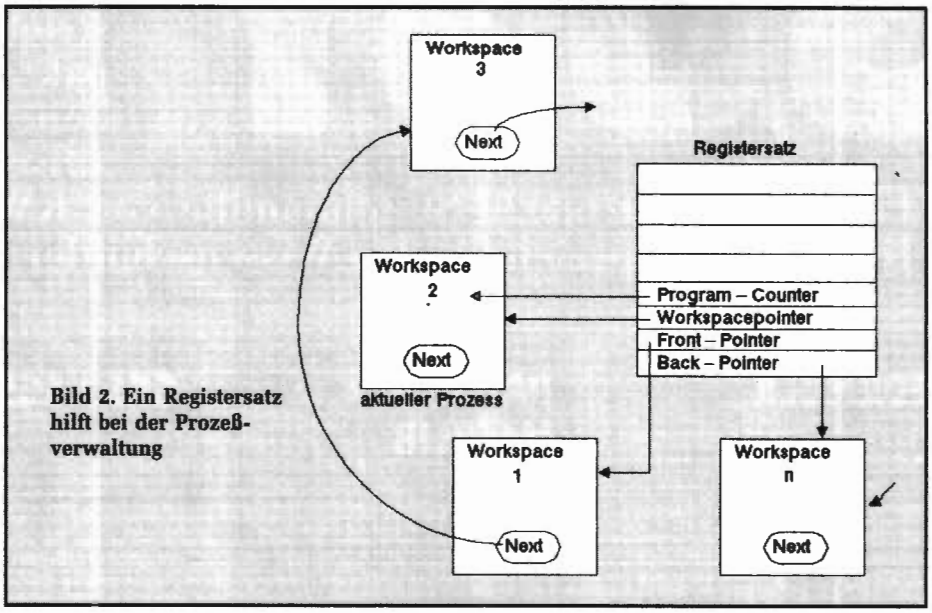


Bild 2. Ein Registersatz hilft bei der Prozeßverwaltung

Über den Bus lassen sich auch Peripherie-Bausteine herkömmlicher Bauart anschließen. Sogar eine DMA- und Interruptlogik (Event-logic) stehen zur Verfügung. Wer also vom Transputerkonzept nicht so begeistert ist, könnte den Chip immer noch als eine sehr leistungsfähige CPU mit eingebauten Schnittstellen mißbrauchen.

Was parallel heißt

Betrachten Sie Prozesse, die gleichzeitig ablaufen und über irgendwelche Kanäle miteinander verbunden sind. Zur Untersuchung der Frage, wie man mehrere Transputer, die je vier Links besitzen, sinnvoll miteinander verschalten könnte, sei ein einfaches (?) Problem betrachtet: Wie schnell kann man einen Vektor von n Zahlen sortieren?
Die schnellsten sequentiellen und allgemeinen Verfahren (Quick- und Heapsort) benötigen dafür in der Größenordnung n * log(n) Schleifendurchläufe. Das, so steht es in den Lehrbüchern, ist die Anzahl der Schritte, die im ungünstigsten Fall nicht unterschritten werden kann. Sequentiell denkende Informatiker glauben also, daß diese Anzahl der Schritte die Schnelligkeit beschränkt. Wenn man allerdings die Sortierarbeit aufteilt, geht es doch schneller. Es gibt sogar eine baumartig aufgebaute Prozeßstruktur, die bei ununterbrochener Anlieferung von immer neuen zu sortierenden Vektoren auf einen durchschnittlichen Sortieraufwand von einem (1 !) Schleifendurchlauf pro Vektor kommt. Doch da dabei eine sehr große Menge von Prozessen verwendet werden müßte, sei dies nur erwähnt. Es sei aber die Aufgabe gestellt, mit Hilfe von n Prozessen einen Vektor von Zahlen in n Schleifendurchläufen zu sortieren. Die Prozesse in *Bild 3* können jeder die Komponenten parallel übernehmen. Nach Übernahme der Werte des Vektors über die Init-Kanäle durchlaufen die Prozesse n Phasen, je n/2 mal den Aktiv- und den Passivzyklus. Dabei beginnen alle ungeraden Prozeßnummern (von links durchgezählt) mit dem Aktivzyklus und alle geraden Prozeßnummern mit dem Passivzyklus. Im Aktivzyklus arbeitet der Prozeß folgende Instruktionen ab:

a) empfange Wert von rechten Nachbarn
b) vergleiche diesen mit gespeichertem Wert
c) sende größeren von beiden an den rechten Nachbarn
d) speichere den kleineren Wert
Im Passivzyklus führt er nur zwei Instruktionen aus:
e) sende gespeicherten Wert an den linken Nachbarn
f) speichere den zurückempfangenen Wert

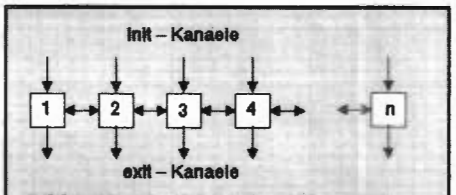


Bild 3. Die Sortierstruktur für einen n-dimensionalen Vektor

Sind diese n Phasen durchlaufen, gibt jeder Prozeß seinen gespeicherten Wert über die Exit-Kanäle weiter – der Vektor ist sortiert. „Logisch" werden die einen sagen, „undurchsichtig" die anderen. In der Tat verlangt das Konzept der verteilten Prozesse andere Denkstrukturen als sie bei sequentiellen Algorithmen notwendig sind. Auch die Algorithmen und Datenstrukturen selbst sehen plötzlich, wie das Beispiel zeigt, ganz anders aus.

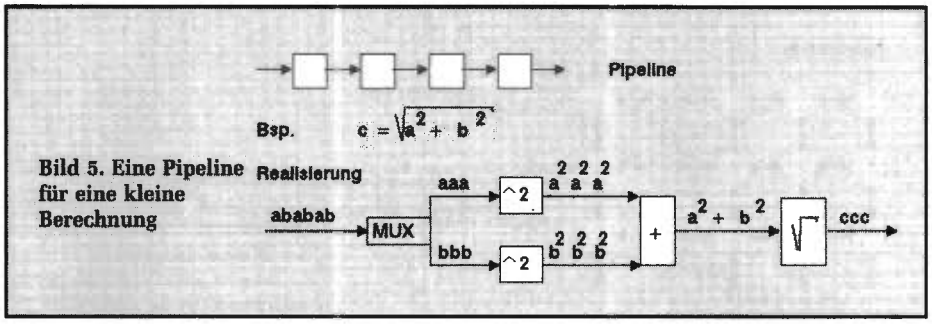


Bild 5. Eine Pipeline für eine kleine Berechnung

Pipelines: Arbeit im Multiplex verschachtelt

Im Zusammenhang mit solche Strukturen werden die Begriffe „Pipeline", „Prozeßarray" oder „Prozeßhypercube" genannt. Das betrifft die Sache mit der Anordnung. Mit voller Absicht soll der Begriff Pipeline hier an eine Röhre voller Öl mit zwei Enden erinnern. An einem Ende kommt genausoviel Öl heraus, wie zur gleichen Zeit am anderen Ende hineingepummpt wird. Trotzdem ist das einzelne Öltröpfchen eine ganze Weile unterwegs – nur merkt man davon nur zu Beginn des Pumpens etwas, falls die Röhre leer war, später nichts, da dann alle Tropfen gleich sind. Bei einer Daten-Pipeline ist das ganz ähnlich: Für manche Probleme läßt sich der Algorithmus, der die Aufgabe löst, als eine Black-Box mit einem Eingang und einem Ausgang betrachten (*Bild 4*). Ein kontinuierlich ankommender Eingabedatenstrom wird von dieser Black-Box in einen kontinuierlich abgehenden Ausgabedatenstrom transformiert. Ist diese Transformation einfach genug, so daß ein Rechner sie in einer Zeit durchführen kann, die kleiner als die Ankunftsrate der Eingabedaten ist, dann kann die Anordnung Schritt halten und die Ergebnisse kommen, nur wenig verzögert, im Anlieferungstakt aus dem Prozeß (der „kurzen" Röhre). Da man aber die Rechenleistung eines Prozesses mit vertretbaren Mitteln nicht beliebig steigern kann, gibt es Schwierigkeiten, sobald diese Zeit-Bedingung nicht mehr erfüllt wird. Falls es jedoch eine Möglichkeit gibt, die Transformation in Untertransformationen aufzuspalten, so daß von den nacheinander auszuführenden Untertransformationen jede nur die von der vorhergehenden Untertransformation berechneten Zwischenergebnisse benötigt, dann läßt sich eine Pipeline konstruieren. Ein einfaches Beispiel dafür sei die mathematische Berechnungsvorschrift C = Wurzel ($A^2 + B^2$).

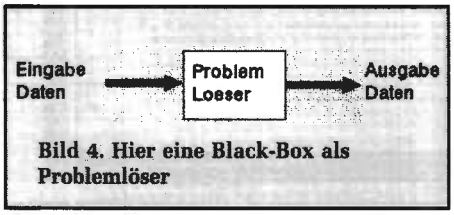


Bild 4. Hier eine Black-Box als Problemlöser

Parallel macht schnell

Werden die Werte für A und B alternierend jeweils mit der Rate t am Eingang angeliefert, so läßt sich gewährleisten, daß mit der Pipeline nach *Bild 5* auch die Ergebnisse C mit der Rate t richtig berechnet abgehen, wenn jede Einzeloperation nicht länger als t zur Durchführung braucht (mit der Ausnahme, daß der Multiplexer in der Zeit t/2 umschalten muß, was bei einer – im Verhältnis zur Wurzel – so einfachen Operation nicht schwer zu garantieren ist). Insgesamt benötigt die Berechnung eines einzelnen C zwar genausoviel Zeit, wie bei einem sequentiellen Algorithmus, jedoch werden immer mehrere Schritte parallel abgearbeitet, so daß die Verarbeitungsrate wesentlich höher ist. Ferner werden die einzelnen Hardware- und Software-Komponenten einfacher und besser ausgelastet. Ein herkömmlicher sequentieller Prozeß würde alle Bearbeitungsschritte hintereinander ausführen müssen, also auch die dafür benötige Hardware besitzen müssen. Er benutzt jedoch pro Zeitperiode immer nur einen geringen Teil dieser Hardware. Der Teilprozeß ↑ 2 in unserem Beispiel muß dagegen nur quadrieren können. Ganz nebenbei entsprechen die bei dieser Art der Paralellisierung entstehenden Konstruktionen dem Ziel der „strukturierten Programmierung".

Strukturen rechnen

Bestimmte Problemstrukturen bieten sich geradezu an, von Transputernetzen aufgenommen zu werden – nämlich immer dann, wenn sie sich in einer logisch/physikalischen Topologie abbilden lassen. Wenn es zum Beispiel um die Simulation von Systemen mit lokaler Interaktion von Subsystemen geht (Ökologie, Zell- und Kristallwachstum), so ist es besonders elegant, dies in Form von entsprechend vernetzten Prozeßarrays zu realisieren, wobei fast allen Prozessen ähnliche, wenn nicht gar gleiche Algorithmen zugrunde liegen. Ein spielerisches, aber typisches Beispiel hierfür ist das „Game of Life" von Conway, ein zellulärer Automat, mit verblüffenden, an lebende Strukturen erinnernden Verhaltensweisen. Die Regeln sind sehr einfach:
Das „Spielfeld" ist eine beliebig große (ideal wäre eine unendliche) Ebene mit Schachbrettmuster. Jedes Feld entspricht einer Zelle, die entweder lebt (logisch 1) oder tot ist (logisch 0). Alle Zellen berechnen ihren nächsten Zustand anhand der momentanen Zustände der Nachbarzellen (8 Nachbarn inkl. Diagonale). Sind zwei Nachbarn lebendig, so ändert sich der Zustand der Zelle nicht. Leben drei Nachbarn, hat die Zelle auf jeden Fall als nächstes den Zustand lebend. Jede andere Zahl von lebenden Nachbarn läßt die Zelle den Zustand tot annehmen. Da diese Berechnungen alle gleichzeitig erfolgen müssen, damit die Zellen nicht ihren Zustand verändern, ehe ihre Nachbarn anhand des momentanen Zustandes entschieden haben, welchen Nachfolgezustand sie annehmen werden, ist ein solches Problem für die Realisierung durch ein synchronisiertes Prozeßarray prädestiniert. Ein Tip – probieren Sie das Game of Life einmal mit einem entsprechenden sequentiellen Algorithmus aus – es entstehen je nach Anfangszustand der Zellen dabei sehr interressante Strukturen (vor allen dann, wenn man lebende und tote Zellen entsprechend als Pixel grafisch auf dem Bildschirm darstellt). Es gibt viele Ideen für die optimale und möglichst universelle Verschaltung paralleler Prozesse. Diese Fragestellung ist vor allen dann Interessant, wenn, wie im Falle der Transputer, diese Verschaltung durch echte „Drähte" erfolgt, die man möglichst selten umstecken möchte. Einige Experten favori-

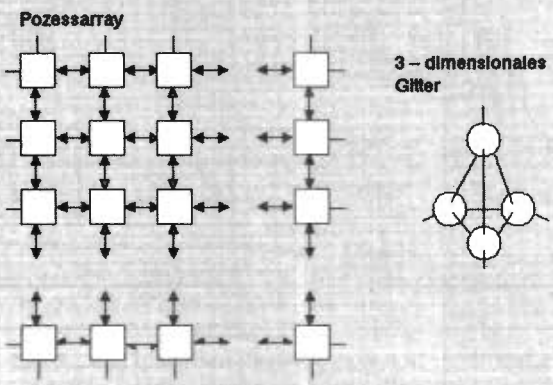


Bild 6. Dreidimensionale Gitter- und Thorus-Struktur. Den Thorus erkennt man, wenn man die offenen Leitungen jeweils mit dem anderen Ende in derselben Spalte (Zeile) verbindet

sieren dabei einen Torus (zum Beispiel die Oberfläche des Schlauches aus einem Autoreifen), andere bevorzugen ein Konstrukt, ähnlich dem Kristallgitter im Diamanten usw. (*Bild 6*). Bis jetzt besteht allerdings noch keine Klarheit darüber, welche Konstruktion in den meisten Fällen einsetzbar ist. Es ist jedenfalls jetzt schon abzusehen, daß Transputer in vielen Gebieten Fuß fassen werden und andere konkurrierende Systeme unter Umständen verdrängen werden. Die Vorteile liegen auf der Hand – und diesmal haben nicht die USA oder Japan die Nase vorn, sondern europäische Firmen.

Desktop Publishing für Einsteiger

Kyocera Deutschland, bekannt durch innovative Laserdrucker, bietet jetzt ein preiswertes Programm an, das die ersten Schritte im Bereich Desktop Publishing erlaubt. News Master, so der Name des Programms, erlaubt das Erstellen von Rundschreiben, Preislisten, Infoblättern, Speisekarten und ähnlichem. Der Vorteil des Programms liegt in seiner recht einfachen Bedienbarkeit, denn der Anwender findet im unteren Bildschirmdrittel Piktogramme für die einzelnen Funktionen. Zusätzlich wird ihm in einer Statuszeile gesagt, was zu tun ist. Die Seite läßt sich in vier verschiedenen Vergrößerungsstufen auf dem Bildschirm darstellen (wahlweise CGA-, EGA- oder Hercules-Karte). Zur Einteilung des Blattes kann man die Seite in bis zu 10 Spalten aufteilen, wahlweise mit einem Block für eine Überschrift oder durchgehend. Die Texteingabe erfolgt mit einem einfachen Texteditor, der über die Cursortasten gesteuert wird. Man kann den Text je nach Bedarf zentrieren oder mit Randausgleich setzen, wobei es keine Trennhilfe gibt. Für die Textgestaltung steht eine ganze Palette von Schriftarten (Fonts) zur Verfügung.

Leider gibt es bei den Schriften weder Umlaute noch „ß" – das Programm kommt aus Amerika und wurde bisher noch nicht eingedeutscht (auch das Handbuch ist in Englisch). Mit einem Fremdeditor erstellte Texte lassen sich als ASCII-Dateien importieren.

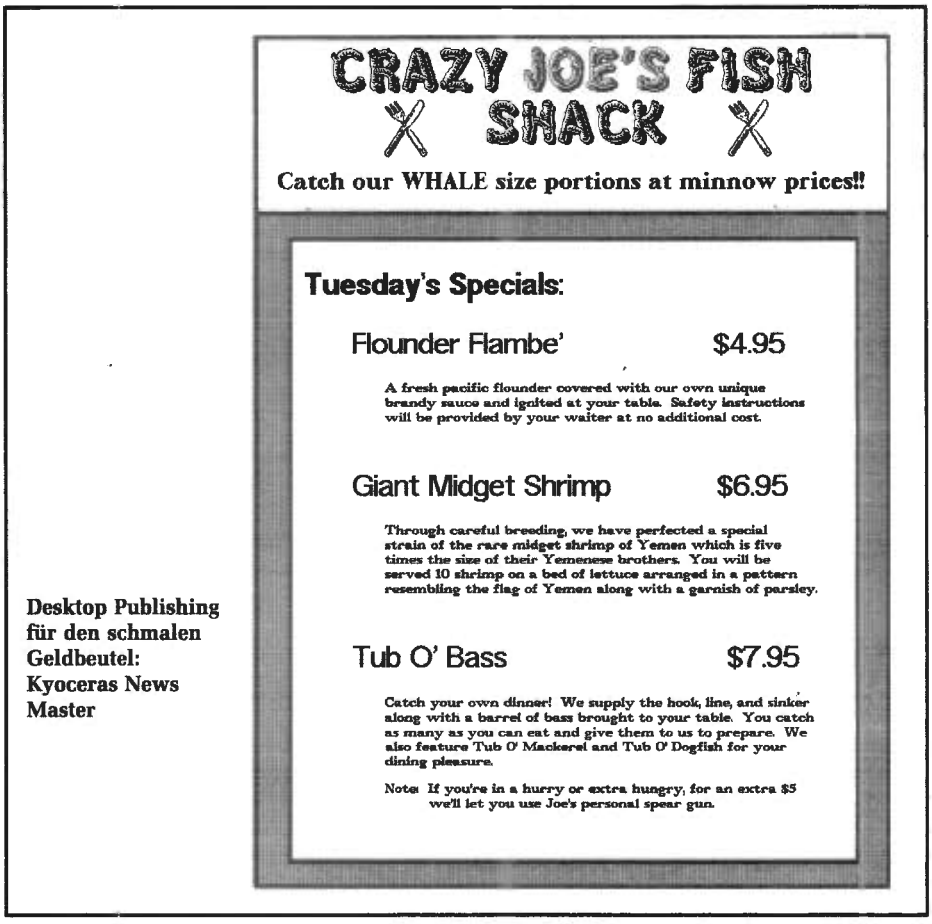
Um den Text ein wenig aufzulockern kann man unter den mitgelieferten Bildern auswählen und das gewünschte Bild in den Text integrieren. Das Erstellen eigener Bilder ist mit News Master nicht möglich.

Das Programm wurde auch zur „Matrixdruckerzeit" erstellt, die Auflösung erreicht nicht die Laserdruckerqualität.

Trotzdem finden sich im umfangreichen Druckermenü der Kyocera F-1010 und der HP-Laserjet+. Das *Bild* vermittelt eine Eindruck der Druckqualität.

Um sich in den Bereich Desktop Publishing einzuarbeiten, ist News Master wegen seines einfachen Aufbaus sicher geeignet; für eine professionelle Nutzung müßten die deutschen Umlaute und ein Grafikeditor hinzugefügt werden.

Jürgen Plate


CRAZY JOE'S FISH SHACK

Catch our WHALE size portions at minnow prices!!

Tuesday's Specials:

Flounder Flambe' $4.95

A fresh pacific flounder covered with our own unique brandy sauce and ignited at your table. Safety instructions will be provided by your waiter at no additional cost.

Giant Midget Shrimp $6.95

Through careful breeding, we have perfected a special strain of the rare midget shrimp of Yemen which is five times the size of their Yemenese brothers. You will be served 10 shrimp on a bed of lettuce arranged in a pattern resembling the flag of Yemen along with a garnish of parsley.

Tub O' Bass $7.95

Catch your own dinner! We supply the hook, line, and sinker along with a barrel of bass brought to your table. You catch as many as you can eat and give them to us to prepare. We also feature Tub O' Mackerel and Tub O' Dogfish for your dining pleasure.

Note: If you're in a hurry or extra hungry, for an extra $5 we'll let you use Joe's personal spear gun.

Desktop Publishing für den schmalen Geldbeutel: Kyoceras News Master

Bernfried Brüggemann

mc-CP/M-Computer mit Uhr

Den Z80-STI-Baustein optimal nutzen

Kern der Uhr ist der Z80-STI-Baustein auf der TERM1-Platine. Er enthält einen im Terminalprogramm nicht benutzten Timer, der bei entsprechender Initialisierung für eine Uhr verwendet werden kann. Ein Turbo-Pascal-Programm lädt die entsprechende Assemblerroutine in den Speicher von TERM1 und initialisiert die Uhr. Die aktuelle Uhrzeit wird von einer Pascal-Prozedur ausgelesen.

Bild 1 zeigt das Listing des Assemblerprogramms, das in den freien Speicherbereich ab Adresse 4700H geladen wird. Die Initialisierungsroutine wird auf 470AH gestartet. Während der Initialisierung sind Interrupts gesperrt. Danach wird das Timer-A-Register laufend mit der um den Faktor 200 vorgeteilten Taktfrequenz am STI-A-Baustein dekrementiert. Beim Nulldurchgang wird das Register auf den Ursprungswert von 192H zurückgesetzt und ein Interrupt ausgelöst. Der STI-A-Baustein liefert das zugehörige Low-Byte der Interruptadresse an den Z80-Prozessor, der aus seinem Interruptregister I das High-Byte dazugibt.

Die Interruptadresse enthält den Zeiger auf die Interruptroutine, die sofort ausgeführt wird.

In der Initialisierungsroutine werden zwei Interrupts zugelassen:

- beim Nulldurchgang des Timers A
- beim Empfang von Daten über die serielle Schnittstelle zum Computer

```
                        .PABS
4700                    .LOC 4700H
                        .PHEX
0060                    ERR=0060H
0063                    GET=0063H
4700    00      STDH:   .BYTE 0H        ;STUNDENZAEHLER,HI-BYTE
4701    00      STDL:   .BYTE 0H        ;LOW-BYTE
4702    00      MINH:   .BYTE 0H        ;MINUTENZAEHLER,HI-BYTE
4703    00      MINL:   .BYTE 0H        ;LOW-BYTE
4704    00      SEKH:   .BYTE 0H        ;SEKUNDENZAEHLER,HI-BYT

4705    00      SEKL:   .BYTE 0H        ;LOW-BYTE
4706    00      NDZ:    .BYTE 0H        ;NULLDURCHGANGSZAEHLER
4707    0000    QRZKOR: .WORD 0H        ;JUSTAGEMARKE
4709    00      PLUMIN: .BYTE 0H        ;FLAG FUER
                                        ;VOR- ODER NACHGEHEN
                                        ;
                ;***************************************************

                ;********  INITIALISIERUNGSROUTINE  ************

                ;***************************************************

470A            INIT:                   ;INITIALISIERUNG
                                        ;VON STI-A
470A    F5              PUSH    PSW
470B    F3              DI
470C    ED5E            IM2             ;INTERRUPTMODUS 2 SETZEN

470E    3E47            MVI     A,47H   ;INTERRUPT-VEKTOREN AUF
                                        ;ADRESSE 47xxH UMBIEGEN
4710    ED47            STAI            ;(OBERES HALBBYTE DER
                                        ;INTERRUPTADRESSE LADEN)

4712    3E77            MVI     A,77H   ;VORTEILER EINSTELLEN
4714    D389            OUT     89H     ;/200
4716    3EC0            MVI     A,0C0H  ;TIMER LADEN /192
4718    D38B            OUT     08BH
471A    3E45            MVI     A,45H   ;POINTER AUF INTERRUPT-
471C    D388            OUT     88H     ;ENABLE-REGISTER RICHTEN

471E    3E30            MVI     A,30H   ;ENABLE INTERRUPT FUER
4720    D387            OUT     87H     ;TIMER A & DATENEMPFANG
4722    D380            OUT     80H
4724    F1              POP     PSW
4725    FB              EI
4726    C9              RET
                        ;
4740                    .LOC 4740H
                ;***************************************************

                ;************ INTERRUPTADRESSEN *****************

                ;***************************************************

4740    0060            .WORD ERR
4742    0060            .WORD ERR
```

Bild 1. Mit dieser Routine wird der Z80-STI-Baustein angesprochen

```
4744    0060                .WORD ERR
4746    0060                .WORD ERR
4748    0060                .WORD ERR
474A    0060                .WORD ERR
474C    0060                .WORD ERR
474E    0060                .WORD ERR
4750    0060                .WORD ERR
4752    0060                .WORD ERR
4754    0060                .WORD ERR
4756    0060                .WORD ERR
4758    0063                .WORD GET       ;ZEICHEN EMPFANGEN
475A    4760                .WORD TA        ;TIMER A NULLDURCHGANG
475C    0060                .WORD ERR
475E    0060                .WORD ERR
                            ;
                            ;
                    ;***********************************************

                    ;* INTERRUPTROUTINE FUER TIMER A NULLDURCHGANG *

                    ;***********************************************

4760    F5          TA:     PUSH    PSW
4761    C5                  PUSH    B
4762    E5                  PUSH    H
4763    ED4B 4707           LBCD    QRZKOR  ;* QUARZKORREKTUR START
4767    03                  INX     B       ;*
4768    21 0000             LXI     H,0     ;* HIER MUSS (IN SEK.)
                                            ;* EINGETRAGEN WERDEN,
                                            ;* NACH WELCHER ZEIT
                                            ;* DIE UNJUSTIERTE UHR
                                            ;* GENAU EINE SEKUNDE
                                            ;* FALSCH GEHT
476B    7D                  MOV     A,L     ;*
476C    B9                  CMP     C       ;*
476D    C2 478D             JNZ     NOKORR  ;*
4770    7C                  MOV     A,H     ;*
4771    B8                  CMP     B       ;*
4772    C2 478D             JNZ     NOKORR  ;*
4775    01 0000             LXI     B,0H    ;*
4778    ED43 4707           SBCD    QRZKOR  ;*
477C    3A 4709             LDA     PLUMIN  ;* PLUMIN=0:
                                            ;* UHR GEHT NACH
477F    FE01                CPI     01H     ;* PLUMIN=1:
4781    CA 4791             JZ      MIN     ;* KEINE JUSTAGE
4784    F2 4794             JP      PLU     ;* PLUMIN=2:
4787    CD 479A             CALL    INCTIM  ;* UHR GEHT VOR
478A    C3 4791             JMP     MIN     ;*
478D    ED43 4707   NOKORR: SBCD    QRZKOR  ;* QUARZKORREKTUR ENDE
4791    CD 479A     MIN:    CALL    INCTIM
4794    E1          PLU:    POP     H
4795    C1                  POP     B
4796    F1                  POP     PSW
4797    FB                  EI
4798    ED4D                RETI
                            ;
```

```
479A                INCTIM:                 ;ZAEHLER INKREMENTIEREN
479A    21 4706             LXI     H, NDZ  ;NULLDURCHGANGS-ZAEHLER
479D    0660                MVI     B,96    ;64 BEI 4.9152 MHz
                                            ;96 BEI 7.3728 MHz
479F    CD 47F2             CALL    INCR
47A2    D8                  RC
47A3    21 4705             LXI     H,SEKL
47A6    060A                MVI     B,0AH
47A8    CD 47F2             CALL    INCR    ;SEKUNDEN-ZAEHLER
47AB    D8                  RC
47AC    21 4704             LXI     H,SEKH
47AF    0606                MVI     B,06H
47B1    CD 47F2             CALL    INCR
47B4    D8                  RC
47B5    21 4703             LXI     H,MINL  ;MINUTEN-ZAEHLER
47B8    060A                MVI     B,0AH
47BA    CD 47F2             CALL    INCR
47BD    D8                  RC
47BE    21 4702             LXI     H,MINH
47C1    0606                MVI     B,06H
47C3    CD 47F2             CALL    INCR
47C6    D8                  RC
47C7    21 4701             LXI     H,STDL  ;STUNDEN-ZAEHLER
47CA    060A                MVI     B,0AH
47CC    CD 47F2             CALL    INCR
47CF    F5                  PUSH    PSW
47D0    3E04                MVI     A,04H   ;UM 24 UHR RESET
47D2    B9                  CMP     C       ;AUF 0 UHR
47D3    C2 47E8             JNZ     NORSET
47D6    3A 4700             LDA     STDH
47D9    FE02                CPI     02H
47DB    C2 47E8             JNZ     NORSET
47DE    3E00                MVI     A,0H
47E0    32 4700             STA     STDH
47E3    32 4701             STA     STDL
47E6    F1                  POP     PSW
47E7    C9                  RET
47E8    F1          NORSET: POP     PSW
47E9    D8                  RC
47EA    3A 4700             LDA     STDH
47ED    3C                  INR     A
47EE    32 4700             STA     STDH
47F1    C9                  RET
                            ;
47F2                INCR:   ;HL=ZAEHLERADRESSE, B=MAX(ZAEHLER)+1
                            ;BEI UEBERLAUF NOCARRY ZURUECK
47F2    4E                  MOV     C,M
47F3    0C                  INR     C
47F4    79                  MOV     A,C
47F5    B8                  CMP     B
47F6    37                  STC
47F7    C2 47FD             JNZ     STORE
47FA    0E00                MVI     C,0H
47FC    3F                  CMC
47FD    71          STORE:  MOV     M,C
47FE    C9                  RET
                            .END
```

Weil der Interrupt beim Nulldurchgang höher priorisiert ist als der beim Datenempfang, wird der Gang der Uhr durch die Datenübertragung nicht beeinflußt. Bei der TERM1-Standardversion mit 7,3728-MHz-Quarz liefert der Timer A 96 Interrupts pro Sekunde. Die Interruptroutine zählt diese Interrupts und setzt sie in die von 4700H bis 4705H gespeicherte aktuelle Uhrzeit um.

Weil die absolute Frequenzgenauigkeit der Computerquarze meistens zu wünschen übrig läßt, kann in der Interruptroutine der Quarzfehler korrigiert werden.

Die Uhr in Pascal

Den Download des Assemblerprogramms in den Speicher von TERM1 übernimmt das Pascalprogramm CLCKINIT in *Bild 2*. Die Voreinstellungen der Konstantendefinition im Vereinbarungsteil und die im Dialog eingegebene Uhrzeit werden bei der Übertragung des Assemblerprogramms in den Hexcode eingefügt. Anschließend wird mit dem WE-Kommando die Initialisierung der Uhr in TERM1 angestoßen.

Bild 3 zeigt das Programm RDCLOCK, das über Grafikbefehle die aktuelle Uhrzeit aus dem TERM1-Speicher abliest. Um den Quarzfehler herauszufinden, läßt man die Uhr zunächst einige Zeit mit der Einstellung PLUMIN='$01' laufen. Aus der Gangabweichung kann man die Zeit berechnen, nach der die Uhr um genau eine Sekunde falsch anzeigt. Diese Zeit wird in Hexcode der Konstanten TIMSEK zugewiesen und PLUMIN='$00' bzw. '$02' gesetzt, je nachdem, ob die Uhr nach- oder vorgeht. Mit dieser Korrektur kann die Gangabweichung der Uhr auf weniger als zwei Sekunden pro Tag begrenzt werden.

```
PROGRAM CLCKINIT;
{initialisiert die uhr in term1}

CONST NDZ:STRING[4]=' $22';
      {der nulldurchgangszaehler erhaelt eine vorgabe, um beim start
      der uhr die zeitverzoegerung durch den download auszugleichen}

      NDPS:STRING[4]=' $60';
      {bei 4.9125 MHz erfolgen 40h nulldurchgaenge des timers a pro sekunde
        "  7.3728 MHz    "     60h          "        "    "    "  "     "   }

      TIMSEK:STRING[4]='0000';
      {die zahl der sekunden in HEX-code, nach der die
      unjustierte uhr genau eine sekunde falsch geht}

      PLUMIN:STRING[4]=' $01';
      {'00' : unjustierte uhr geht nach
       '01' : es erfolgt keine korrektur des quarzfehlers
       '02' : unjustierte uhr geht vor}

VAR STDH,STDL,MINH,MINL,SEKH,SEKL,TSH,TSL:STRING[4];
    TIME:STRING[8];
    AS:STRING[1];


BEGIN
WRITELN('Bitte die Uhrzeit (hh:mm:ss) angeben:');
READLN(TIME);
STDH:=' $0'+COPY(TIME,1,1);
STDL:=' $0'+COPY(TIME,2,1);
MINH:=' $0'+COPY(TIME,4,1);
MINL:=' $0'+COPY(TIME,5,1);
SEKH:=' $0'+COPY(TIME,7,1);
SEKL:=' $0'+COPY(TIME,8,1);
TSH:=' $'+COPY(TIMSEK,1,2);TSL:=' $'+COPY(TIMSEK,3,2);

{die voreinstellungen und korrekturen werden in die download-daten eingetragen}

WRITELN(^[^['GWD $4700',STDH,STDL,MINH,MINL,SEKH);
WRITELN(^[^['GWD $4705',SEKL,NDZ,' $00 $00',PLUMIN);
WRITELN(^[^['GWD $470A $F5 $F3 $ED $5E $3E $47 $ED $47 $3E $77 $D3 $89');
WRITELN(^[^['GWD $4716 $3E $C0');
WRITELN(^[^['GWD $4718 $D3 $8B $3E $45 $D3 $88 $3E $30 $D3 $87 $D3 $80');
WRITELN(^[^['GWD $4724 $F1 $FB $C9');
WRITELN(^[^['GWD $4740 $60 $00 $60 $00 $60 $00 $60 $00 $60 $00 $60 $00');
WRITELN(^[^['GWD $474C $60 $00 $60 $00 $60 $00 $60 $00 $60 $00 $60 $00');
WRITELN(^[^['GWD $4758 $63 $00 $60 $47 $60 $00 $60 $00 $F5 $C5 $E5 $ED');
WRITELN(^[^['GWD $4764 $4B $07 $47 $03 $21',TSL,TSH,' $7D $B9 $C2 $8D $47');
WRITELN(^[^['GWD $4770 $7C $B8 $C2 $8D $47 $01 $00 $00 $ED $43 $07 $47');
WRITELN(^[^['GWD $477C $3A $09 $47 $FE $01 $CA $91 $47 $F2 $94 $47 $CD');
WRITELN(^[^['GWD $4788 $9A $47 $C3 $91 $47 $ED $43 $07 $47 $CD $9A $47');
WRITELN(^[^['GWD $4794 $E1 $C1 $F1 $FB $ED $4D $21 $06 $47 $06',NDPS,' $CD');
WRITELN(^[^['GWD $47A0 $F2 $47 $D8 $21 $05 $47 $06 $0A $CD $F2 $47 $D8');
WRITELN(^[^['GWD $47AC $21 $04 $47 $06 $06 $CD $F2 $47 $D8 $21 $03 $47');
WRITELN(^[^['GWD $47B8 $06 $0A $CD $F2 $47 $D8 $21 $02 $47 $06 $06 $CD');
WRITELN(^[^['GWD $47C4 $F2 $47 $D8 $21 $01 $47 $06 $0A $CD $F2 $47 $F5');
WRITELN(^[^['GWD $47D0 $3E $04 $B9 $C2 $E8 $47 $3A $00 $47 $FE $02 $C2');
WRITELN(^[^['GWD $47DC $E8 $47 $3E $00 $32 $00 $47 $32 $01 $47 $F1 $C9');
WRITELN(^[^['GWD $47E8 $F1 $D8 $3A $00 $47 $3C $32 $00 $47 $C9 $4E $0C');
WRITELN(^[^['GWD $47F4 $79 $B8 $37 $C2 $FD $47 $0E $00 $3F $71 $C9');

WRITE(^[^['GWE $470A'^M);

WRITELN(^[^['AUhr in TERM1 ist initialisiert.');
END.
```

Bild 2. Mit CLCKINIT wird die Uhr initialisiert

```
PROGRAM RDCLOCK;
{liest die uhr in term1 ab}

TYPE STRING8=STRING[8];

PROCEDURE RDTIME(VAR TIME:STRING8);
VAR A:CHAR;AS:STRING[1];IO:TEXT;

BEGIN
ASSIGN(IO,'KBD:');RESET(IO);
TIME:='  :  :  ';
WRITE(^[^['GWF $4700'^M);READ(IO,A);
STR(ORD(A),AS);TIME[1]:=AS[1];
WRITE(^[^['GWF $4701'^M);READ(IO,A);
STR(ORD(A),AS);TIME[2]:=AS[1];
WRITE(^[^['GWF $4702'^M);READ(IO,A);
STR(ORD(A),AS);TIME[4]:=AS[1];
WRITE(^[^['GWF $4703'^M);READ(IO,A);
STR(ORD(A),AS);TIME[5]:=AS[1];
WRITE(^[^['GWF $4704'^M);READ(IO,A);
STR(ORD(A),AS);TIME[7]:=AS[1];
WRITE(^[^['GWF $4705'^M);READ(IO,A);
STR(ORD(A),AS);TIME[8]:=AS[1];
WRITE(^[^['A');
END;


VAR TIME:STRING8;

BEGIN
RDTIME(TIME);
WRITELN(TIME);
END.
```

Bild 3. RDCLOCK ist ein Minimalprogramm zum Ablesen der Uhr

Hubert Nehring

SCEDIT

Ein Bildschirmeditor für den mc-68000-Computer unter CP/M-68k

Beim Umgang mit dem etwas abenteuerlich zu bedienenden Editor ED.68k entsteht schnell der Wunsch, die Möglichkeiten des Monitorprogramms – das sich ja immer noch unter dem CP/M verborgen hält – auch von diesem aus zu nutzen. Dies wird vom vorliegenden Assembler-Programm realisiert.

Das Programm arbeitet mit der Monitorversion 1.4 und einer IBM-Tastatur zusammen. Durch Löschen des CP/M-Flags ($FFE96F) wird der virtuelle Textspeicher des Monitors wieder aktiviert und so die Möglichkeit zum Editieren auf dem Bildschirm geschaffen. Zu beachten ist dabei allerdings, daß dieser Textspeicher unter CP/M normalerweise als Druckerspooler verwendet wird. Man sollte also während des Editierens auf das Ausdrucken von Texten verzichten oder den Spooler auf eine andere Adresse legen (ab 512K).

Zur Vereinfachung des Programms wurde auf die Verarbeitung überlanger Zeilen verzichtet, was für einen Programmeditor sicher nicht stört. Das Programm ist für den Betrieb im 80-Zeichen-Modus gedacht.

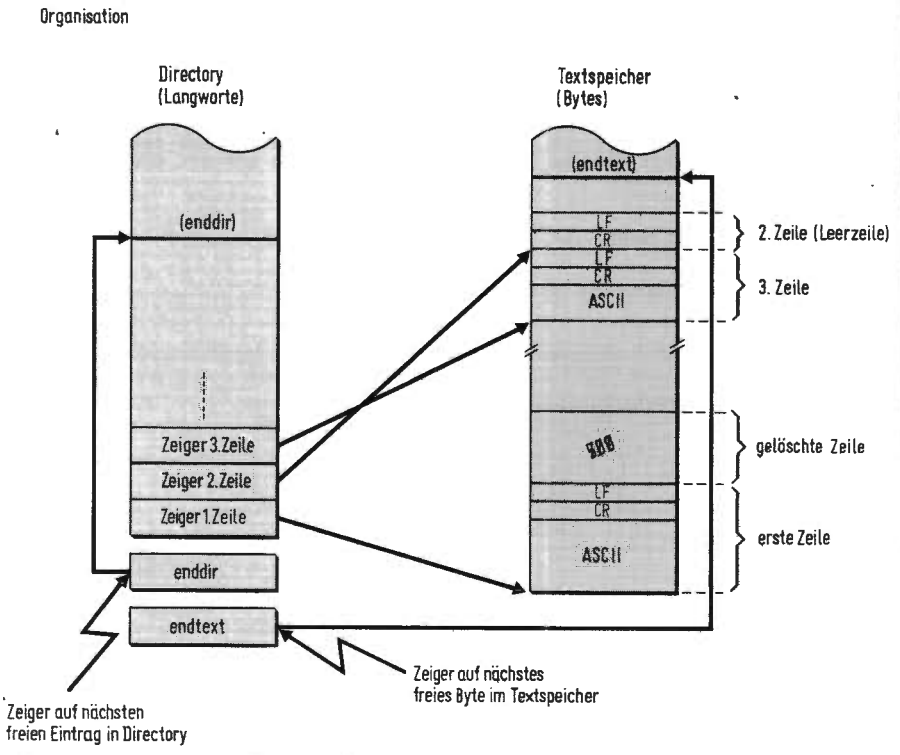


Bild 1. So funktioniert die String-Verwaltung

Aufruf

Der Aufruf geschieht wie bei ED.68k mit SCEDIT <Dateiname>, wobei bei schon vorhandenen Textdateien auch Wildcards erlaubt sind. SCEDIT lädt nun die Datei (falls vorhanden) und ändert die Typ-Extension in „BAK", damit im Notfall die alte Datei noch greifbar ist. Dann wird das Bild aufgebaut, das aus folgenden Teilen besteht:

- 4 Kopfzeilen, die zur Anzeige des Dateinamens, des Menüs und der Tabulatorpositionen dienen.
- Ein 20 Zeilen umfassendes Textfenster, in dem der Text bearbeitet werden kann.
- Die Statuszeile des Betriebssystems.

Falls eine schon vorhandene Datei aufgerufen wurde, werden im Textfenster die ersten 20 Zeilen angezeigt. Man kann nun wie gewohnt mit dem Cursor herumfahren und Zeichen einfügen, überschreiben, löschen usw., wobei der Editor ohne Zeilennummern auskommt.

Das Ordnungskriterium ist vielmehr die y-Position des Cursors im Textfenster. Es stehen folgende Kommandos zur Verfügung, die über den Cursorblock, Controlcodes oder Escape-Sequenzen eingegeben werden:

Pfeiltasten: Die Steuerung des Cursors. Der Cursor bewegt sich nur bis eine Zeile unter den Text. Ist der untere Rand des Textfensters erreicht, wird gescrollt.

Home/End: Der Cursor wird nach links oben bzw. links unten gesetzt.

PgUp/PgDn: Es wird eine Seite (19 Zeilen) zurück bzw. weiter geblättert.

Ins: Umschalten in den Einfügungs-Modus

Backspace: Zeichen löschen

TAB/SHIFT-TAB: Tabulator

Return: Übernahme der Zeile in den Textspeicher. Nur mit Return abgeschlossene Zeilen werden auch wirklich übernommen.

CTRL T: Tabulatorposition setzen bzw. löschen. Die Tabulatorpositionen wer-

```
*****************************************************************
esc:      .equ $1b              * ESC
clrs:     .equ $0c              * clear screen
cr:       .equ $0d              * carrige return
lf:       .equ $0a              * line feed
pgup:     .equ $0c              * page up
pgdn:     .equ $18              * page down
cuup:     .equ $1e              * cursor up
cudn:     .equ $1f              * Cursor down
culeft:   .equ $1d              * cursor left
curight:  .equ $1c              * curdor right
cuhome:   .equ $0b              * Cursor home
cuend:    .equ $11              * Cursor end
ins:      .equ $10              * insert character
tab:      .equ $09              * Tabulator
revtab:   .equ $1a              * Tabulator rueckwaerts
backsp:   .equ $08              * Backspace
ctrl_r:   .equ $12
ctrl_t:   .equ $14

eof:      .equ $1a              * End of File

bdos:     .equ 2                * BDOS - Trap
dirconio: .equ 6                * direct consol i/o
prtstr:   .equ 9                * print string
openfile: .equ 15
closefile:.equ 16
searchfi: .equ 17               * Search for first
deletef:  .equ 19               * Delete File
readsequ: .equ 20               * File sequentiell lesen
writesequ:.equ 21               * File sequentiell schreiben
makefile: .equ 22               * File kreieren
renamef:  .equ 23               * File umbenennen
currdisk: .equ 25               * return current disk

setwindow:.equ $ff7400          * Textfenster setzen
monram:   .equ $ffe000          * Anfang des Monitor RAM's
line3:    .equ $ff80f0          * Basisadresse der 3.Bildschirmzeile

linesize: .equ 4096             * maximale Anzahl der Zeilen
textsize: .equ 80000            * Groesse des Textspeichers
*****************************************************************
* Hauptprogramm                                                 *
*****************************************************************
          .text

SCEDIT:   movea.l $4(a7),a6     * BASE PAGE Pointer nach A6
          bsr     scinit        * alles initialisieren
          tst.b   d0            * Initialisferungsfehler ?
          beq     mainloop      *   nein -; Einstieg Hauptschleife
scende:   rts                   * EXIT --; zurueck zu CP/M
mainloop: move.w  #$ff,d1       * Zeichen von Console holen
          bsr     dodconio
          cmp.b   #$20,d0       * ist es ein Steuerzeichen ?
          blt     comsel        *   ja -; Befehl auswaehlen
          tst.b   esc_flag      * wurde vorher ESC gedrueckt ?
          beq     output        *   nein -; Zeichen einfach ausgeben
          clr.b   esc_flag
comsel:   lea     comtab,a0     * A0 = Zeiger auf die Befehlstabelle
          move.w  (a0)+,d2      * Anzahl der Befehle - 1 holen
```

Bild 2. Das Editor-Programm benutzt viele Routinen des mc-68000-Monitors

```
comloop:  cmp.b   (a0,d2.w),d0  * Befehl in Tabelle gefunden ?
          bne     nextcom       *   nein -; weiter suchen
          lea     comvec,a0     *   ja -; A0 = Sprungtabellen-Zeiger
          asl.w   #2,d2         * D2 enthaelt die Nr. des Befehls
          movea.l (a0,d2.w),a1  * Befehlsvektor nach A1 laden und...
          jmp     (a1)          * Befehl anspringen
nextcom:  dbra    d2,comloop
          bra     mainloop      * Befehl war nicht vorhanden
***** OUTPUT  Zeichen in D0 einfach ausgeben **********************
output:   move.b  d0,d1         * kein Befehl --; dann Textzeichen...
          bsr     dodconio      * ausgeben
          bra     mainloop
***** EXIT    aus dem Editor aussteigen, uninit ********************
exit:     bsr     writefile     * File zurueckschreiben
          move.l  #$00004e17,d0 * alte Zustaende wiederherstellen
          jsr     setwindow
          lea     rescue,a0     * Monitor-Flags auf alten Zustand
          move.l  (a0)+,$940(a3)
          move.l  (a0)+,$944(a3)
          move.w  (a0)+,$948(a3)
          move.b  (a0)+,$7fa(a3)
          move.b  (a0)+,$96f(a3)
          bsr     clearsc
          bra     scende
***** DOESC   esc_flag setzen *************************************
doesc:    st      esc_flag
          bra     mainloop
***** RETURN  Zeile in Textspeicher uebernehmen *******************
creturn:  bsr     texttest      * Textspeicher voll ?
          tst.b   d0
          bmi     texterr       *    ja -; Fehlermeldung ausgeben
          bsr     getline       * Bildschirm-Zeile zum Input-Puffer
          bsr     linein        * Input-Puffer-Zeile zum Textspeicher
          cmpi.b  #23,$7f9(a3)  * muss gescrollt werden ?
          bne     cret1         *   nein -; cret1
          addq.l  #1,d6         *   ja -; korrigiere Zeiger
          addq.l  #1,d5
          bsr     crlf          * jetzt cr und lf ausgeben
          bsr     compdir
          cmpa.l  enddir,a4     * gibt es die naechste Zeile schon ?
          bge     cretend       *   nein -; fertig
          bsr     disline       *   ja -; Zeile anzeigen
          bra     cretend
cret1:    bsr     crlf
cretend:  bra     mainloop
***** PAGEUP  eine Seite hoeher ***********************************
pageup:   bsr     clearsc
          cmp.l   #20,d5        * oberes Ende des Textes erreicht ?
          bhi     pgup1
          moveq   #1,d5         *    ja -; Textanfang anzeigen
          bra     pgup2
pgup1:    sub.l   #19,d5        *    nein -; 19 Zeilen zurueck gehen
pgup2:    bsr     dispage       *    und die Seite anzeigen
          bra     mainloop
***** PAGEDOWN       eine Seite tiefer ****************************
pagedown: bsr     clearsc
          add.l   #19,d5        * 19 Zeilen weiter gehen und...
          move.l  d5,d6
          bsr     compdir
          cmpa.l  enddir,a4     * schauen, ob Textende erreicht.
          blt     pgdn1         *   nein -;
          sub.l   #19,d5        *   ja -; D5-Wert wiederherstellen
pgdn1:    bsr     dispage       * Seite anzeigen
          bra     mainloop
```

```
***** Initialisierungsroutine ***************************************
scinit:   movem.l initreg,d5/d6/a2-a5   * initialisiere Zeiger
          lea     $5c(a6),a0      * 1. FCB
          lea     $80(a6),a1      * default DMA
          tst.b   (a1)            * Filename vorhanden ?
          beq     initerr1
          tst.b   (a0)            * default Disk ?
          bne     contini1
          move.w  #currdisk,d0    * ja, hole default Disk-Nr.
          trap    #bdos
          addq.b  #1,d0           * korrigiere Disk-Nr. fuer FCB
          move.b  d0,(a0)         * Disk-Nr. nach FCB
contini1: move.l  a0,d1
          move.w  #searchfi,d0    * File im Directory suchen
          trap    #bdos
          tst.b   d0              * File gefunden ?
          bmi     newfile
          asl.w   #5,d0           * ja...
          addq.w  #1,d0
          adda.w  d0,a1
          addq.w  #1,a0
          moveq   #10,d2
fcbcopy1: move.b  (a1)+,(a0)+     * Filename nach FCB
          dbra    d2,fcbcopy1
          bsr     readfile        * File lesen
          bsr     renfile         * File umbenennen
          bra     contini2
newfile:  lea     $5c(a6),a0      * neues File...
          moveq   #11,d2          * keine Wildchar. im Filenamen
wildtest: cmpi.b  #'?',(a0)+
          beq     initerr2
          dbra    d2,wildtest
contini2: move.l  #txt1,d1        * Kopfzeile ausgeben
          bsr     strout
          lea     $5c(a6),a0      * jetzt den Filenamen ausgeben...
          move.b  (a0)+,d1        *   erst die Laufwerkkennung
          add.b   #$40,d1
          bsr     dodconio
          move.b  #':',d1         *   dann ein ':'
          bsr     dodconio
          moveq   #7,d2           *   jetzt der Filename (8 Zeichen)
fcblp1:   move.b  (a0)+,d1
          cmp.b   #' ',d1         *   skip spaces
          beq     fcbend1
          bsr     dodconio
fcbend1:  dbra    d2,fcblp1
          move.b  #'.',d1         *   dann ein '.'
          bsr     dodconio
          moveq   #2,d2           *   und der Filetyp (3 Zeichen)
fcblp2:   move.b  (a0)+,d1
          bsr     dodconio
          dbra    d2,fcblp2
          move.l  #txt2,d1        * jetzt Menuezeilen ausgeben
          bsr     strout
          move.l  #$00044e17,d0   * Daten fuer Textfenster
          jsr     setwindow       * Textfenster setzen
          lea     rescue,a0       * Monitor-Flags retten + initialis.
          move.l  $940(a3),(a0)+  *   rette TAB-Positionen
          move.l  $944(a3),(a0)+
          move.w  $948(a3),(a0)+
          move.l  #$01010101,$940(a3)   * setze Standard-TABs
          move.l  #$01010101,$944(a3)
          move.w  #$0101,$948(a3)
          move.b  $7fa(a3),(a0)+  *   rette Linkflag
          move.b  #$00,$7fa(a3)   *   keine ueberlangen Zeilen
          move.b  $96f(a3),(a0)+  *   CP/M-Flag retten
          move.b  #$00,$96f(a3)   *   Textspeicher freigeben
          moveq   #8,d2           * TAB-Markierungen anzeigen
          moveq   #8,d0
markloop: bsr     togpfeil
          addq.l  #8,d0
          dbra    d2,markloop
          bsr     dispage         * erste Seite anzeigen
          clr.w   d0              * kein Initialisierungsfehler
endinit:  rts

initerr1: move.l  #txt6,d1        * Initialisierungsfehler
          bra     errout
initerr2: move.l  #txt7,d1
errout:   bsr     strout
          move.b  #$ff,d0         * setze Fehlerflag
          bra     endinit
***** Ausgabe von CR und LF *****************************************
crlf:     move.l  #cr_lf,d1
          bsr     strout
          rts
***** Ausgabe von Clear Screen **************************************
clearsc:  moveq   #clrs,d1
          bsr     dodconio
          rts
***** direct consol i/o ausfuehren **********************************
dodconio: move.w  #dirconio,d0
          trap    #bdos
          rts
***** print string ausfuehrem ***************************************
strout:   move.w  #prtstr,d0
          trap    #bdos
          rts

* Editorfunktionen ******* diese Funktionen enden mit BRA mainloop****

***** eine Leerzeile einfuegen  *************************************
insertl:  bsr     texttest        * Textspeicher voll ?
          tst.b   d0
          bmi     texterr         *   ja -; Fehlermeldung
          move.l  #esc_i,d1       * ESC I ausgeben (insert line)
          bsr     strout
          clr.l   d6              * berechne Zeilennummer in der der
          move.b  $7f9(a3),d6     * Cursor steht
          subq.l  #4,d6
          add.l   d5,d6
          bsr     compdir         * berechne Directoryeintr.-Adresse
          bsr     dirup           * Directory nach oben bewegen
          movea.l endtext,a5      * und Leerzeile bestehend aus...
          move.l  a5,(a4)
          move.b  #cr,(a5)+       * cr und...
          move.b  #lf,(a5)+       * lf...
          move.l  a5,endtext      * in den Textspeicher schreiben
          bra     mainloop

texterr:  move.l  #txt8,d1        * Fehlermeldung: Textspeicher voll
          bsr     strout
          bra     mainloop
***** eine Zeile loeschen *******************************************
deletel:  clr.l   d6              * berechne Zeile in der der
          move.b  $7f9(a3),d6     * Cursor steht
          move.b  d6,-(a7)        * rette Cursorposition
          subq.l  #4,d6
```

```
          add.l   d5,d6
          bsr     compdir
          cmpa.l  enddir,a4       * ist ueberhaupt Text vorhanden ?
          bge     nodel           * nein, kein delete
          bsr     clroldstr       * Zeile im Textspeicher loeschen
          bsr     dirdown
          move.l  #esc_j,d1       * ESC J/Cursor-Home ausgeben
          bsr     strout          * (clear end of screen)
          bsr     dispage
nodel:    move.b  (a7)+,$7f9(a3)
          move.l  #esc_s,d1       * ESC S (Cursor zum Zeilenanfang)
          bsr     strout          * ausgeben, um Cursor zu setzen
          bra     mainloop
****** Cursor-down   achtet auf Textende und scrollt ggf. ************
curdown:  clr.l   d6              * berechne Dirctoryeintr.-Adresse
          move.b  $7f9(a3),d6     * der naechsten Zeile und ...
          subq.l  #3,d6
          add.l   d5,d6
          bsr     compdir
          cmpa.l  enddir,a4       * pruefe, ob sie existiert.
          bgt     curdoend        * nein -; kein Cursor-down
          cmpi.b  #23,$7f9(a3)    * Cursor in der letzten Zeile ?
          beq     curdo1          * ja -; springe, es wird gescrollt
          move.w  #cudn,d1        * nein -; Cursor-down ausgeben
          bsr     dodconio
          bra     curdoend
curdo1:   bsr     crlf            * scrollen
          addq.l  #1,d5           * Zeiger korrigieren
          cmpa.l  enddir,a4       * noch eine Zeile im Textspeicher ?
          bge     curdoend        *   nein -; fertig
          bsr     disline         *   ja -; Zeile ausgeben
curdoend: bra     mainloop
***** angezeigte Seite ueberpruefen *********************************
verify:   bsr     clearsc
          bsr     dispage
          bra     mainloop
***** zum Textanfang gehen ******************************************
top:      move.l  #1,d5           * D5 = 1.Zeile
          bsr     clearsc
          bsr     dispage
          bra     mainloop
***** zum Textende gehen ********************************************
bottom:   move.l  enddir,d5       * Textende suchen
          sub.l   #diranf,d5
          lsr.l   #2,d5
          sub.l   #18,d5
          cmp.l   #1,d5           * mindestens 20 Zeilen im Speicher ?
          ble     top             *   nein -; top ausfuehren
          bsr     clearsc         *   ja -; letzte Seite anzeigen
          bsr     dispage
          move.w  #cuend,d1
          bsr     dodconio
          bra     mainloop
***** charsel   toggelt Zeichensatz *********************************
charsel:  tst.b   charflag        * teste charflag
          beq     chars1          *  $00 -; auf deutschen Zeichens.
          move.l  #esc_r0,d1      *  $ff -; auf ASCII-Zeichensatz
          clr.b   charflag
          bra     chars2
chars1:   move.l  #esc_r2,d1
          st      charflag
chars2:   bsr     strout
          bra     mainloop
```

```
***** doctrl_t: Tabulatorposition setzen/loeschen und anzeigen *******
doctrl_t: move.b  #ctrl_t,d1
          bsr     dodconio
          clr.l   d0
          move.b  $7f8(a3),d0     * Cursorposition x holen
          bsr     togpfeil        * Markierung toggeln
          bra     mainloop
***** togpfeil   Tabulatormarkierung toggeln  D0.L: Position *********
togpfeil: movea.l a2,a0
          adda.l  d0,a0
          tst.b   $1000(a0)       * ist eine Markierung da ?
          beq     setpfeil        *   nein -; setze eine Markienung
          clr.b   $1000(a0)       *   ja -; loesche die Markierung
          clr.b   $1800(a0)
          clr.b   $2000(a0)
          clr.b   $2800(a0)
          bra     togpfend
setpfeil: lea     pfeiltab,a1     * Markierung setzen
          move.b  (a1)+,$1000(a0) * Bitmasken aus Tab. in Kopfzeile
          move.b  (a1)+,$1800(a0) * eintragen.
          move.b  (a1)+,$2000(a0)
          move.b  (a1)+,$2800(a0)
togpfend: rts
***** File lesen ***************************************************
readfile: lea     $5c(a6),a0      * lese Textfile gemaess 1. FCB...
          move.l  a0,d1
          move.w  #openfile,d0
          trap    #bdos
          move.l  a5,(a4)+        * nach Textspeicher...
readloop: move.l  a0,d1
          move.w  #readsequ,d0    * sequentiell...
          trap    #bdos
          lea     $80(a6),a1      * ueber default DMA-Puffer.
          moveq   #$7f,d2
readdma:  cmpi.b  #eof,(a1)       * EOF erreicht ?
          beq     close           *   ja -; close
          move.b  (a1)+,(a5)
          cmpi.b  #lf,(a5)+       * Ende einer Zeile erreicht ?
          bne     contread        *   nein -; weiter
          move.l  a5,(a4)+        *   ja -; naechste Zeile eintr.
contread: dbra    d2,readdma
          bra     readloop
close:    move.l  a0,d1
          move.w  #closefile,d0
          trap    #bdos
          move.l  a5,endtext      * endtext und enddir setzen
          subq.l  #4,a4
          move.l  a4,enddir
          rts
***** File in .BAK umbenennen ***************************************
renfile:  lea     $38(a6),a0
          moveq   #8,d2
fcbcopy2: move.b  $24(a0),(a0)+   * Filenamen ohne Extender nach 2.FCB
          dbra    d2,fcbcopy2
          move.b  #'B',(a0)+      * 2. FCB bekommt Type .BAK
          move.b  #'A',(a0)+
          move.b  #'K',(a0)+
          lea     $38(a6),a0
          move.l  a0,d1           * loesche vorhandenes .BAK File
          move.w  #deletef,d0
          trap    #bdos
          moveq   #11,d2
fcbcopy3: move.b  (a0),$34(a0)    * 1. FCB fuer rename-Funktion
          addq.l  #1,a0           * vorbereiten
```

```
          dbra    d2,fcbcopy3
          lea     $5c(a6),a0
          move.l  a0,d1           * File umbenennen
          move.w  #renamef,d0
          trap    #bdos
          rts
***** Text auf Diskette zurueckschreiben ****************************
writefile:cmpi.l  #diranf,enddir * ist ueberhaupt ein Text vorhanden
          beq     nowritef       * nein, dann kein writefile
          lea     $5c(a6),a0     * Zeiger auf 1.FCB
          clr.b   12(a0)         * Parameter in 1.FCB loeschen
          clr.b   32(a0)
          move.l  a0,d1
          move.w  #makefile,d0   * File kreieren
          trap    #bdos
          lea     diranf,a4
          movea.l (a4)+,a5       * text aus Textspeicher ueber...
writeloop:lea     $80(a6),a1     * DMA-Puffer nach File schreiben
          moveq   #$7f,d2
writedma: move.b  (a5)+,(a1)
          cmpi.b  #1f,(a1)+
          bne     contwrite
          movea.l (a4)+,a5
          cmpa.l  enddir,a4      * Textende erreicht ?
          bhi     filleof        * ja, springe
contwrite:dbra    d2,writedma
          move.l  a0,d1
          move.w  #writesequ,d0  * sequentiell schreiben.
          trap    #bdos
          bra     writeloop
filleof:  move.b  #eof,(a1)+     * Record mit EOF auffuellen
          dbra    d2,filleof
          move.l  a0,d1
          move.w  #writesequ,d0  * und noch schreiben
          trap    #bdos
          move.l  a0,d1
          move.w  #closefile,d0  * dann File schliessen
          trap    #bdos
nowritef: rts

* Routinen zur Textbearbeitung ***************************************
***** hole Textzeile aus dem Input-Puffer in den Textspeicher ********
linein:   bsr     compdir        * Adresse des Directoryeintrags
          cmpa.l  enddir,a4      * Zeile anhaengen oder ueberschr. ?
          bge     anfueg         * anfuegen -; springe
          bsr     clroldstr      * alte Textzeile loeschen
          bra     line1
anfueg:   addq.l  #4,enddir      * anfuegen -; neuer Directoryeintrag
line1:    movea.l endtext,a5
          move.l  a5,(a4)
          lea     linebuff,a0    * Zeilen-Puf. nach Textsp. bis ...
inloop:   move.b  (a0)+,(a5)
          cmpi.b  #cr,(a5)+      * CR erreicht ist.
          bne     inloop
          move.b  #1f,(a5)+      * dann LF anfuegen
          move.l  a5,endtext
          rts
**** Adresse des Directoryeintrags berechnen ************************
compdir:  move.l  d6,d1          * Umrechnung Zeilennummer(D6) in die
          subq.l  #1,d1          * Adresse des Direktoryeintrags(A4)
          asl.l   #2,d1
          add.l   #diranf,d1
          movea.l d1,a4
          rts
```

```
**** Zeile im Textspeicher loeschen *********************************
clroldstr:movea.l (a4),a5        * Zeilen-Adresse aus Directory holen
clrloop:  move.b  (a5),d0
          clr.b   (a5)+          * Zeichen loeschen
          cmp.b   #1f,d0         * war es vorher ein LF ?
          bne     clrloop        *   nein -; naechst. Zeichen loeschen
          rts                    *   ja -; fertig
**********************************************************************
dirup:    addq.l  #4,enddir      * bewegt Directory ab A4 um einen
          move.l  enddir,d2      * Eintrag nach oben
          movea.l d2,a0
          sub.l   a4,d2
          lsr.l   #2,d2          * D2: Zahl zu kopierend. Eintr. - 1
diruploop:move.l  -(a0),4(a0)
          dbra    d2,diruploop
          rts
**********************************************************************
dirdown:  move.l  enddir,d2      * bewegt Directory ab A4 um einen
          sub.l   a4,d2          * Eintrag nach unten
          lsr.l   #2,d2
          subq.l  #1,d2
dirdnloop:move.l  4(a4),(a4)+
          dbra    d2,dirdnloop
          subq.l  #4,enddir
          rts
**********************************************************************
getline:  lea     $7d0(a3),a0    * Zeile vom Bildschirm nach Inbuff
          clr.w   d0
linkloop: move.b  $7f9(a3),d1    * suche Cursor-Pos. in Linktabelle
          cmp.b   (a0,d0.w),d1
          beq     getline1       * gefunden ? ja, springe
          addq.w  #1,d0
          bra     linkloop
getline1: mulu    #80,d0         * D0 = Offset fuer Bildschirmspeicher
          lea     linebuff,a0    * Bildschirmspeichers
          clr.l   d6             * berechne Zeilennummer fuer linein
          move.b  d1,d6          * D6 = Cursorposition
          subq.l  #4,d6          * Korrektur wegen Kopfzeilen
          add.l   d5,d6          * D6 = D6 + Nr. der obersten Zeile
          moveq   #79,d2
getloop:  move.b  (a3,d0.w),(a0)+    * hole ganze Zeile in den Puffer
          addq.w  #1,d0
          dbra    d2,getloop
getline2: cmp.b   #' ',-(a0)     * Ende der Zeile suchen und...
          beq     getline2
          move.b  #cr,$1(a0)     * CR anfuegen.
          rts
***** eine Zeile auf dem Bildschirm anzeigen ************************
disline:  movea.l (a4),a5        * Directoryeintrag nach Textzeiger
disline1: move.b  (a5)+,d1       * Text ab Zeiger lesen
          cmp.b   #1f,d1         * Zeilenende erreicht ?
          beq     disline2       *   ja -;
          bsr     dodconio       *   nein -; Zeichen ausgeben
          bra     disline1
disline2: cmpi.b  #23,$7f9(a3)   * Cursor am unteren Bildschirmende ?
          beq     enddisli       * ja -; kein LF
          bsr     dodconio
enddisli: rts
********** eine Bildschirmseite anzeigen ***************************
dispage:  moveq   #19,d2         * eine Bildschirmseite = 20 Zeilen
          move.l  d5,d6          * D5 enthaelt Nr. der obersten Zeile
disploop: bsr     compdir        * Umrechnung Zeilennr. -; Dir-Zeiger
          cmpa.l  enddir,a4      * Directoryende erreicht ?
```

```
          bge     dispend         *   ja -; fertig
          bsr     disline         *   nein Zeile anzeigen
          addq.l  #1,d6           * naechste Zeile
          dbra    d2,disploop     * schon 20 Zeilen ausgegeben ?
dispend:  moveq   #cuhome,d1      *   ja -; Cursor Home
          bsr     dodconio
          rts
****Garbage Collection ********************************************
garbcoll: lea     txtmem,a5       * Anzahl der Bytes im Textspeicher
          move.l  endtext,d1      * berechnen und...
          sub.l   a5,d1
          subq.l  #1,d1           * als Zaehler verwenden
garbloop: tst.b   (a5)+           * suche $00 im Textspeicher
          bne     garbcont        *   nicht gefunden -; weiter suchen
          bsr     collect         *   gefunden -; Collection
garbcont: dbra    d1,garbloop
          rts
collect:  move.l  a5,-(a7)        * Adresse des Strings retten
          subq.l  #1,a5
          movea.l a5,a1
coloop1:  tst.b   (a1)+           * Laenge des Strings bestimmen
          beq     coloop1
          move.l  endtext,d2
          sub.l   a1,d2
          subq.l  #1,d2
coloop2:  move.b  (a1)+,(a5)+     * und den Textspeicher runterkopieren
          dbra    d2,coloop2
          suba.l  a5,a1
          movea.l (a7)+,a5
          move.l  a1,d2
          move.l  enddir,d3
          sub.l   #diranf,d3      * Directory-Laenge berechnen und...
          lsr.l   #2,d3
          subq.l  #1,d3           * als Zaehler verwenden
          lea     diranf,a4       * jetzt alle Zeiger mit Werten
coloop3:  cmpa.l  (a4)+,a5        * groesser A5 korrigieren
          bhi     nocorr
          sub.l   d2,-4(a4)
nocorr:   dbra    d3,coloop3
          sub.l   d2,d1
          sub.l   d2,endtext
          rts

texttest: cmpi.l  #txtmem+textsize,endtext   * Platz im Textspeicher?
          bls     textok
          bsr     garbcoll        * sonst Garbage-Collection
          cmpi.l  #txtmem+textsize,endtext   * nochmal testen
          bls     textok          * ist jetzt Platz im Speicher ?
          move.b  #$ff,d0
          bra     ttestend
textok:   clr.b   d0              * ja, alles ok
ttestend: rts
*******************************************************************
          .data
          .even
***** Tabelle mit Kommandos ***************************************
comtab:   .dc.w   22              * Anzahl-1 der Kommandos
          .dc.b   cr,esc,pgup,pgdn,cuup,cudn
          .dc.b   culeft,curight,cuhome,cuend
          .dc.b   ins,backsp,tab,revtab,ctrl_t,ctrl_r
          .dc.b   "eidbbvc"
          .even
***** Tabelle mit Sprungzielen der Kommandos **********************
comvec:   .dc.l   creturn,doesc,pageup,pagedown,output,curdown
          .dc.l   output,output,output,output,output,output
          .dc.l   output,output,doctrl_t,output,exit,insertl
          .dc.l   deletel,top,bottom,verify,charsel
*******************************************************************
charflag: .dc.b   $00             * Flag fuer Zeichensatz 00=US, $ff=dt
esc_flag: .dc.b   $00             * Flag fuer Escapesequemz
***** Editor Texte *************************************************
txt1:     .dc.b   clrs,esc,$34
          .dc.b   "  S C E D I T   V1.3   |                 "
          .dc.b   "                |                HN -- 10/86 "
          .dc.b   esc,"y",$3b,$20,"File: $"
txt2:     .dc.b   esc,$35
          .dc.b   esc,"y",$20,$21,esc,$34,"CTRL+",esc,$35,"    "
          .dc.b   esc,$34,"t",esc,$35,":set/clr tab        "
          .dc.b   esc,$34,"r",esc,$35,":toggle keybd   "
          .dc.b   esc,"y",$65,$21,esc,$34,"Cursorblock",esc,$35
          .dc.b   esc,"y",$20,$22,esc,$34,"ESC",esc,$35,"+     "
          .dc.b   esc,$34,"i",esc,$35,"nsert/"
          .dc.b   esc,$34,"d",esc,$35,"elete-line   "
          .dc.b   esc,$34,"t",esc,$35,"op/"
          .dc.b   esc,$34,"b",esc,$35,"ottom of text   "
          .dc.b   esc,$34,"v",esc,$35,"erify   "
          .dc.b   esc,$34,"c",esc,$35,"har.sel   "
          .dc.b   esc,$34,"e",esc,$35,"xit   "
          .dc.b   esc,"y",$20,$23
          .dc.b   "                                                "
          .dc.b   "                                                $"
txt6:     .dc.b   cr,lf,"Error: Filename fehlt.",cr,lf,"$"
txt7:     .dc.b   cr,lf,"Keine Wildchards bei neuem File.",cr,lf,"$"
txt8:     .dc.b   clrs,esc,"Y** Textspeicher voll ! $"

esc_r0:   .dc.b   esc,"R0$"
esc_r2:   .dc.b   esc,"R2$"
esc_i:    .dc.b   esc,"I$"
esc_j:    .dc.b   esc,"J",cuhome,"$"
esc_s:    .dc.b   esc,"S$"
cr_lf:    .dc.b   cr,lf,"$"
****** Werte zur Initialisierung der Register *********************
          .even
initreg:  .dc.l   $1              * D5
          .dc.l   $1              * D6
          .dc.l   line3           * A2
          .dc.l   monram          * A3
          .dc.l   diranf          * A4
          .dc.l   txtmem          * A5
***** Tabelle mit Werten fuer die Tabulatormarkierung ************
pfeiltab: .dc.b   $fe,$7c,$38,$10

* Arbeitsspeicher fuer den Editor  *********************************
          .even
endtext:  .dc.l   txtmem      * Zeiger auf Ende des Textes
enddir:   .dc.l   diranf      * Zeiger auf Ende des Directory's

          .bss
          .even
rescue:   .ds.b   $20         * Platz fuer Monitor-Flags
linebuff: .ds.b   $100        * Zeilenpuffer z. B. fuer Eingabe
          .even
diranf:   .ds.l   linesize          * Platz fuer Directory
txtmem:   .ds.b   textsize+100      * Platz fuer Text und etwas Reserve


          .even
          .end
```


mc-soft

Verwendung der Register

D0,D1	Übergabe z. B. bei BDOS-Aufrufen
D2.x	Verwendung als Zähler
D5.L	Nummer der obersten Textfenster-Zeile
D6.L	Nummer der aktuellen Zeile
A0,A1	Zeiger zur allgemeinen Verwendung
A2	Zeiger auf die 4. Kopfzeile zur Anzeige der TAB-Positionen
A3	Zeiger auf Monitor-RAM-Basisadresse
A4	Zeiger auf Eintrag im Directory
A5	Zeiger auf Byte im Textspeicher
A6	Zeiger auf die BASE-Page

den in der Kopfzeile durch kleine Pfeile gekennzeichnet.

CTRL R: wechselt zwischen deutscher und amerikanischer Tastenbelegung

ESC i: Einfügen einer Leerzeile über dem Cursor.

ESC d: Löschen der Zeile, in der der Cursor steht.

ESC t: Erste Seite anzeigen.

ESC b: Letzte Seite anzeigen. Sind weniger als 20 Zeilen im Textspeicher, wird ESC t ausgeführt.

ESC v: Baut die gerade angezeigte Bildschirmseite neu auf. Hilft bei Unklarheiten und nach Fehlermeldungen.

ESC c: wechselt zwischen deutschem und amerikanischem Zeichensatz

ESC e: Beendet eine Editor-Sitzung und schreibt den Inhalt des Textspeichers unter dem alten Dateinamen auf die Diskette zurück.

Internes

Der Textspeicher wird über ein Verfahren verwaltet, das durch die Adressierungsarten des 68000 (postincrement, predecrement) gut unterstützt wird (vgl.[1]). Dieses Verfahren (*Bild 1*) ähnelt ein bißchen der Stringverwaltung von Microsoft Basic-Interpretern und arbeitet folgendermaßen:
Der Arbeitsspeicher des Editors besteht aus einem sogenannten Directory, dem eigentlichen Textspeicher und zwei Zeigern, die jeweils auf das Ende dieser beiden Speicherblöcke zeigen – im Programmtext „enddir" und „endtext" genannt.

Das Directory besteht aus Zeigern, die jeweils auf eine Zeile im Textspeicher zeigen, dabei soll der erste Zeiger auf die erste Textzeile zeigen, der zweite auf die zweite Zeile usw. Im Textspeicher sind die Textzeilen einfach hintereinander angeordnet, wobei jede Zeile durch ein CR/LF abgeschlossen ist. Dieses Zeilenvorschub-Zeichen wird daher auch als Endmarkierung für eine Zeile verwendet.
Obwohl der Editor für den Benutzer ohne explizite Zeilennummern arbeitet, werden zur einfacheren Verwaltung intern die Directory-Einträge (= Zeilen) von Eins an durchnummerriert.
Das Verfahren soll nun anhand elementarer Editorfunktionen beschrieben werden.

Zeile einfügen

Zuerst wird die Nummer des Zeigers bestimmt, der auf die einzufügende Zeile zeigen soll. Diese ergibt sich aus der Position des Cursors auf dem Bildschirm. Man muß dazu nur die y-Position des Cursors (steht im Monitorram [3]) und die Nummer der ersten Zeile im Textfenster (steht in D5 s. u.) kennen und addieren, wobei noch eine Korrektur gemacht werden muß, weil das Textfenster erst in der 4. Zeile beginnt. Aus diesem Wert läßt sich dann die Adresse des Zeigers im Directory berechnen. Dann wird der Directoryinhalt über diesem Zeiger um eine Position nach oben verschoben und „enddir" entsprechend korrigiert. Der Zeiger wird mittels „endtext" auf das Ende des Textspeichers gesetzt, wo der Text eingetragen wird. Nun muß nur noch „endtext" angepaßt werden.

Zeile überschreiben

Wieder muß zuerst aus der Nummer der Zeile, in der der Cursor steht, die Adresse des Zeigers im Directory bestimmt werden. Der String, auf den der Zeiger im Augenblick zeigt, wird gelöscht. Dann wird der Zeiger auf das Ende des Textspeichers gesetzt, wo der neue String angefügt wird. Nun muß noch „endtext" angepaßt werden.

Zeile löschen

Zuerst wird wieder die Adresse des Zeigers bestimmt, der auf die Zeile zeigt, die gelöscht werden soll. Anschließend muß noch der Zeiger aus dem Directory entfernt werden, indem der Directoryinhalt über dem Zeiger um eine Position nach unten geschoben wird. Zum Schluß muß noch „enddir" korrigiert werden.
Durch häufiges Einfügen und Löschen von Zeilen sammelt sich im Textspeicher ziemlich viel Müll (Bytes, die $00 enthalten) an. Um also ein Überlaufen des Textspeichers zu vermeiden, muß bei Erreichen der oberen Grenze des Textspeichers eine sogenannte Garbage Collection durchgeführt werden. Diese Routine sucht im Textspeicher nach Nullen und räumt sie aus dem Speicher. Der Text kann nachrücken, so daß wieder Platz am oberen Ende des Textspeichers frei wird. Natürlich müssen die Zeiger im Directory sowie „endtext" entsprechend korrigiert werden. Sollte trotz dieser Maßnahme kein Platz im Textspeicher geschaffen werden können, wird eine Fehlermeldung ausgegeben.
Das Programm (*Bild 2*) verwendet den Default-DMA-Puffer und die Default-FCBs, die CP/M-68k zur Verfügung stellt [2].

Änderungen und Erweiterungen:

Die Größe des Textspeichers ist im vorliegenden Programm auf 80000 Zeichen festgesetzt worden, kann aber durch Umdefinieren des Labels „textsize" beliebig vergrößert werden. Die maximale Anzahl der Directory-Einträge (= maximale Zeilenzahl) ist auf 4096 festgelegt, was wohl für die meisten Anwendungen ausreicht. Sie kann aber durch Ändern des Labels „linesize" auf einen beliebigen Wert gesetzt werden. Achtung: Ein eventuelles Überschreiten dieser maximalen Zeilenzahl wird vom Programm nicht überprüft.
Es ist sehr leicht möglich, zusätzliche Kommandos (z. B. append, find) zu implementieren. Dazu muß der Kommandozähler in „comtab" um eins erhöht, der ASCII-Wert des Kommandocodes an „comtab" angehängt und die Adresse der Routine in „comvec" angefügt werden. Vieleicht sollten dann auch die Menüzeilen angepaßt werden.
Zum Schluß sei nochmals darauf hingewiesen, das SCEDIT keine überlangen Zeilen verarbeiten kann. Man hüte sich davor, es zu versuchen.

Literatur

[1] *Schollwöck, Ulrich:* 68000, 68008 Maschinensprache, iwt-Verlag, München, 1984

[2] CP/M-68k Programmer's Guide, Digital Research

[3] Handbuch des mc-68000-Computers, DSM Digital-Service

Martin N. Annen

Vier Bildschirmseiten in Turbo-Pascal

CGA-Karten optimal nutzen

In Turbo-Pascal (Version 3.0) geschriebene Programme geben ihre Meldungen unabhängig von der aktiven Bildschirmseite immer auf Bildschirmseite 0 aus. Mit geringen Modifikationen können Turbo-Pascal-Programme auf CGA-und AGA-Karten vier unabhängige Bildschirmseiten ansprechen.

Viele werden sich schon gefragt haben, wie Turbo-Pascal-Programme Zeichen auf den Bildschirm ausgeben.

Um zu sehen, wie die Bildschirmausgabe erfolgt, genügt bereits ein kleines Programm:

Begin Write('Sahnetorte') End.

Dieses Programm wird mit der Option 'C' auf Diskette (test.com) compiliert.

```
DOS 3.1 A>type tab_1.asc
-s 0100 3000 cd 10
1397:0229
1397:024D
1397:0257
1397:027F
1397:02A7
1397:02C1
1397:02D6
1397:032F
1397:033C
1397:0363
1397:03FA
1397:0404
1397:040A
1397:0446
1397:0470
1397:047A
1397:04D5
1397:08D8
1397:08E8
1397:0918
1397:091F
1397:2EF8
-q

DOS 3.1 A>
```

Bild 1. Mit dem Debugger lassen sich die INT-10H-Aufrufe schnell finden

Der erzeugte Code läßt sich mit dem Debugger (Eingabe: debug test.com) näher betrachten. Da nur bestimmte Funktionen des BIOS-Interrupt 10H die Wahl der Bildschirmseite erlauben, durchsuchen wir nun mit der Suchfunktion des Debuggers zunächst das Programm nach Stellen, an denen dieser Interrupt ausgelöst wird:

Eingabe: s 0100 3000 cd 10
(cd 10 ist die hexadezimale Zeichenfolge, die in Maschinensprache den Interrupt 10H auslöst.)

Suchen mit dem Debugger

Wir erhalten eine Liste von 21 (evtl. auch mehr) Adressen, an denen sich die gesuchte Zeichenfolge befindet (*Bild 1*). Diese Stellen werden der Reihe nach mit dem Debugger disassembliert (Eingabe: u 0229 usw.). Den disassemblierten Quelltext kann man daraufhin leicht untersuchen, ob der Interrupt 10H angesprochen wird und ob es eine Funktion mit Wahl der Bildschirmseite im Textmodus ist.

Hier sollen exemplarisch nur zwei dieser Stellen gemeinsam betrachtet werden.

Eingabe: u 08D4 08E9
Wir erhalten die in *Bild 2* gezeigte Ausgabe.

Im Speicher mit der Adresse 0BA9:08D4H steht der Befehl: „Lade das AH-Register mit 0EH". Das ist die Funktion: „Ein Zeichen ausgeben und Cursor weiterbewegen". In der nächsten Programmzeile wird das BH-Register durch eine Exklusiv-Oder-Verknüpfung mit sich selbst auf Null gesetzt. Der Inhalt des BH-Registers bestimmt beim Auslösen des Interrupts 10H die aktive Bildschirmseite.

Auch nach dem Befehl unter Adresse 0BA9:08DDH, der das AH-Register mit 09H lädt (Funktion: „Zeichen und Attribut an Cursor-Position schreiben"), wird das BH-Register auf Null gesetzt. Mit den beiden folgenden Befehlen wird die Anzahl der auszugebenden Zeichen in das CX-Register und das Zeichenattribut in das BL-Register geladen, bevor der Interrupt ausgelöst wird. Im PC-Sonderheft [1] werden weitere BIOS-Funktionen erläutert.

Die Analyse aller übrigen Speicherstellen in *Bild 1* ergibt, daß an insgesamt sechs Stellen eine Funktion mit Angabe der Bildschirmseite aufgerufen wird (z. B. auch „Cursor positionieren", „Cursorposition abfragen").

Was ist eine Laufzeit-Bibliothek

Mit Ausnahme der zuletzt betrachteten Stelle wird immer unmittelbar vor der Anweisung INT 10H die Seite 0 mit dem Befehl XOR BH,BH zur aktuellen Seite gemacht.

```
-u 08d4 08e9
0BA9:08D4 B40E          MOV     AH,0E
0BA9:08D6 32FF          XOR     BH,BH
0BA9:08D8 CD10          INT     10
0BA9:08DA EB45          JMP     0921
0BA9:08DC 52            PUSH    DX
0BA9:08DD B409          MOV     AH,09
0BA9:08DF 32FF          XOR     BH,BH
0BA9:08E1 B90100        MOV     CX,0001
0BA9:08E4 8A1E0800      MOV     BL,[0008]
0BA9:08E8 CD10          INT     10
-q
```

Bild 2. In der Turbo-Pascal-Laufzeit-Bibliothek wird die Bildschirmseite 0 aktiviert

Alle erwähnten Funktionsaufrufe erfolgen aus der Laufzeit-Bibliothek von Turbo-Pascal. Eine Laufzeit-Bibliothek ist die Sammlung von Maschinensprache-Routinen, die der Turbo-Pascal-Compiler jedem compilierten Programm voranstellt.

Um unser Problem zu lösen, müssen alle Aufrufe, die für die Auswahl der Bildschirmseite relevant sind, mit dem Befehl zum Laden des BH-Registers mit der gewünschten Ausgabeseite ergänzt werden (z. B. MOV BH,AUSGABE_SEITE). Dieser Befehl beansprucht jedoch mehr Speicherplatz als mit dem Streichen des Befehls XOR BH,BH frei wird. An einer geeigneten Stelle im Speicher muß daher eine kurze Unterroutine stehen, die das Laden übernimmt und den Interrupt auslöst. Außerdem benötigt man eine Speicherstelle, in der die jeweils aktuelle Ausgabeseite steht.

Lösung auf Maschinen-Ebene

Innerhalb der Laufzeit-Bibliothek bietet sich für die Unterroutine der Bereich ab der Offset-Adresse 0107H an. Alle erforderlichen Änderungen im Maschinen-Code sind in *Bild 3* zu sehen. Unter der Offset-Adresse 107H ist die Nummer der Ausgabeseite abspeichert. Zu Beginn sollte stets die Ausgabe auf Seite 0 erfolgen. Die Routine, die bei Offset-Adresse 0108H beginnt, lädt das BH-Register mit der Nummer der jeweilig gewünschten Ausgabeseite und löst den Interrupt 10H aus. Danach geht es mit dem normalen Ablauf der Laufzeit-Bibliothek weiter.

Dementsprechend sind in den jeweiligen Stellen des normalen Codes die beiden Befehle XOR BH,BH und INT 10H durch CALL 0108 und NOP zu ersetzen.

Die unter den Offset-Adressen 08E1H und 08E4 stehenden MOV-Befehle müssen verschoben werden, damit auch hier der richtige Ersatz durch den CALL-Aufruf erfolgen kann.

Um diese Code-Änderungen praktisch durchzuführen, bieten sich zwei verschiedene Lösungen an:

– Änderung am Objektcode
– Turbo-Pascal-Include-Datei

```
-u 0108 010f

1398:0108 2E            CS:
1398:0109 8A3E0701      MOV     BH,[0107]
1398:010D CD10          INT     10
1398:010F C3            RET
-u 0255 0258

1398:0255 E8B0FE        CALL    0108
1398:0258 90            NOP
-u 02a5 02a8

1398:02A5 E860FE        CALL    0108
1398:02A8 90            NOP
-u 0361 0364

1398:0361 E8A4FD        CALL    0108
1398:0364 90            NOP
-u 08d6 08d9

1398:08D6 E82FF8        CALL    0108
1398:08D9 90            NOP
-u 08df 08e9

1398:08DF B90100        MOV     CX,0001
1398:08E2 8A1E0800      MOV     BL,[0008]
1398:08E6 E81FF8        CALL    0108
1398:08E9 90            NOP
-u 091d 0920

1398:091D E8E8F7        CALL    0108
1398:0920 90            NOP
-q
```

Bild 3. Die Laufzeit-Bibliothek wird gepatcht

```
{ Include-File SCRPG_4.bib}

VAR  ScrPg: byte absolute cseg:$0107;

PROCEDURE Show_ScrPg (nr: byte);
VAR reg: RECORD
           al,ah: Byte;
           bx,cx,dx,bp,si,di,ds,es,flags: Integer;
         END;
BEGIN
  reg.ah := 5;
  IF nr IN [0..3] THEN BEGIN
    reg.al := nr; intr($10,reg) END;
END;

PROCEDURE Output_ScrPg (nr: byte);
BEGIN
  IF nr IN [0..3] THEN ScrPg := nr
END;

PROCEDURE mod_runtime_lib;
BEGIN
  mem[cseg:$0107] := $00;   { Byte, Nr. der Ausgabeseite}
  mem[cseg:$0108] := $2e;   { CS: }
  mem[cseg:$0109] := $8a;
  mem[cseg:$010a] := $3e;
  mem[cseg:$010b] := $07;
  mem[cseg:$010c] := $01;   { MOV BH,[0107] }
  mem[cseg:$010d] := $cd;
  mem[cseg:$010e] := $10;   { INT 10 }
  mem[cseg:$010f] := $c3;   { RET }
  mem[cseg:$0255] := $e8;
  mem[cseg:$0256] := $b0;
  mem[cseg:$0257] := $fe;   { CALL 0108 }
  mem[cseg:$0258] := $90;   { NOP }
  mem[cseg:$02a5] := $e8;
  mem[cseg:$02a6] := $60;
  mem[cseg:$02a7] := $fe;   { CALL 0108 }
  mem[cseg:$02a8] := $90;   { NOP }
  mem[cseg:$0361] := $e8;
  mem[cseg:$0362] := $a4;
  mem[cseg:$0363] := $fd;   { CALL 0108 }
  mem[cseg:$0364] := $90;   { NOP }
  mem[cseg:$08d6] := $e8;
  mem[cseg:$08d7] := $2f;
  mem[cseg:$08d8] := $f8;   { CALL 0108 }
  mem[cseg:$08d9] := $90;   { NOP }
  mem[cseg:$091d] := $e8;
  mem[cseg:$091e] := $e8;
  mem[cseg:$091f] := $f7;   { CALL 0108 }
  mem[cseg:$0920] := $90;   { NOP }
  mem[cseg:$08df] := $b9;
  mem[cseg:$08e0] := $01;
  mem[cseg:$08e1] := $00;   { MOV CX, 0001 }
  mem[cseg:$08e2] := $8a;
  mem[cseg:$08e3] := $1e;
  mem[cseg:$08e4] := $08;
  mem[cseg:$08e5] := $00;   { MOV BL, [0008] }
  mem[cseg:$08e6] := $e8;
  mem[cseg:$08e7] := $1f;
  mem[cseg:$08e8] := $f8;   { CALL 0108 }
  mem[cseg:$08e9] := $90;   { NOP }
END;
```

Bild 4. Turbo-Pascal patcht die Laufzeit-Bibliothek selbst

Objektcode ändern

Die einfachste Lösung besteht darin, den Objektcode des compilierten Programms mit dem Debugger zu ändern. Bild 3 zeigt den geänderten Objektcode. Im eigentlichen Pascal-Programm sind dann nur noch zwei kleine Unterroutinen zum Schalten der jeweiligen Ausgabe- bzw. sichtbaren Bildschirmseite erforderlich.

Nach dem Aufruf des Debuggers mit „debug prgname.com" wird die Speicherstelle $0107 mit dem Befehl „e 0107" auf Null gesetzt. Mit dem Befehl a (assemble) können Sie die Assembler-Befehle (Mnemoniks) aus Bild 3 direkt eingeben. Selbstverständlich können die Änderungen auch mit e und Eingabe der jeweiligen hexadezimalen Werte vorgenommen werden. Anschließend wird das geänderte Programm mit dem Befehl w des Debuggers wieder auf Diskette zurückgeschrieben.

Pascal-Include-Datei

Wem die Manipulationen mit dem Debugger nicht ganz geheuer sind, oder wer nach einer einfachen Lösung des Problems innerhalb eines Pascal-Programms sucht, der findet in der in *Bild 4* (SCRPG_4.BIB) dargestellten Include-Datei alles erforderliche, um in seinen Programmen mehrere Bildschirmseiten zu benutzen.

In der Prozedur mod_runtime_lib werden genau die Modifikationen im Maschinencode vorgenommen, wie sie im vorherigen Kapitel dargestellt wurden.

Dies geschieht, indem einmal zu Beginn des Hauptprogramms diese Prozedur aufgerufen wird. Durch den Befehl mem werden dann Byte für Byte die entsprechenden Speicherzellen geändert. Wer die Modifikationen mit dem Debugger vorgenommen hat, benötigt diese Prozedur und den Aufruf natürlich nicht. Um nun mit mehreren Bildschirmseiten zu arbeiten, muß sowohl die jeweils sichtbare Seite als auch die Seite, auf die der Text ausgegeben wird, bestimmt werden können. Dies geschieht durch die beiden anderen kleinen Routinen und die Variable ScrPg.

Die Speicherzelle 0107H (enthält die Nummer der Bildschirmseite für die Textausgabe) ist durch die absolute Deklaration der Variablen ScrPg zugeordnet, so daß direkt darauf zugegriffen werden kann.

```
{$I ScrPg_4.bib}

VAR a, aa : Integer;
    b     : Char;

BEGIN
  mod_runtime_lib; a := 0; ClrScr;
  GotoXY(32,12); Write('Das ist Seite 0 ');
  Output_ScrPg(1);
  GotoXY(8,20); Write('Das ist Seite 1 ');
  Output_ScrPg(2);
  GotoXY(63,5);  Write('Das ist Seite 2 ');
  Output_ScrPg(3);
  GotoXY(50,18); Write('Das ist Seite 3 ');
  REPEAT
    output_ScrPg(a);
    GotoXY(40,25); Write('Welche Seite ?    (Ende = ESC) ');
    read(kbd,b);
    IF b IN ['0'..'3',#27] THEN VAL(b,a,aa);
    show_ScrPg(a);
  UNTIL b = #27;
  Output_ScrPg(0); Show_ScrPg(0);
END.
```

Bild 5. Ein Testprogramm beschreibt vier Bildschirmseiten

In der Prozedur Output_Scrpg wird die Variable ScrPg (gibt die gewünschte Bildschirmseite an) gesetzt.

Auf die gewünschte Bildschirmseite schaltet die Prozedur Show_ScrPg um. Die Variable nr gibt die gewünschte Bildschirmseite an. Mit der Pascal-Prozedur Intr wird durch den Aufruf der Funktion 5 des BIOS-Interrupt 10H die aktive Bildschirmseite umgeschaltet.

Textausgabe im Hintergrund

Alle Turbo-Pascal-Funktionen und -Prozeduren für die Bildschirmsteuerung können bei Textausgabe auf die gerade sichtbare Bildschirmseite verwendet werden. Es ist sogar möglich, im Hintergrund Text auf eine andere Seite auszugeben. Dabei muß beachtet werden, daß die Prozeduren ClrScr, ClrEol, DelLine, InsLine und ein evtl. Bildschirm-Scrollen stets auf dem gerade sichtbaren Bildschirm wirken und somit während einer Textausgabe im Hintergrund nicht verwendet werden können. Da im Hintergrund aufgebaute Bildschirme aber im allgemeinen eine klar definierte Maske aufweisen und keine Textbewegungen erfordern, ist das kein Problem. Nur ein gegebenenfalls zu Beginn nötiges Löschen des nicht sichtbaren Bildschirmes (oder eines Fensters!) muß dadurch korrekt ausgeführt werden, daß für diesen Zweck der Ausgabebildschirm mit Show_ScrPg (nr der Seite des Ausgabebildschirms) kurz sichtbar gemacht wird.
In *Bild 5* sehen Sie ein einfaches Programm zur Demonstration der neuen Funktionen. Nachdem der Code der Laufzeit-Bibliothek durch die Prozedur mod_runtime_lib geändert wurde, wird auf alle vier Bildschirmseiten 'Das ist Seite x' geschrieben. Danach kann man sich die vier Seiten nach Eingabe der betreffenden Ziffer (in der Repeat-Schleife) beliebig ansehen. Nach Verlassen der Schleife mit ESC werden dann wieder Ausgabebildschirm und sichtbarer Bildschirm auf Null geschaltet. Experimentieren Sie mit diesem Programm, um den Umgang mit mehreren Bildschirmseiten in Turbo-Pascal zu erproben.

Turbo-Editor schreibt auf Seite 0

Wer mit den genannten Funktionen und insbesondere auch mit dem Debug-Kommando experimentiert, sollte wissen, daß der Editor des Turbo-Pascal-Systems direkt in die Seite 0 schreibt. Beim Experimentieren geschieht es daher leicht, daß man den Turbo-Editor nicht von Seite 0 aus aufruft und dessen Textausgaben nicht mehr sieht. Falls Programme die beschriebenen Funktionen verwenden, sollten Sie vor Beendigung stets die sichtbare Bildschirmseite und die Seite für die Ausgabe wieder auf Null setzten.

Literatur

[1] *Schmidt, Hans-Jürgen:* IBM-Bios-Funktionen. Das PC-Sonderheft. Seite 8. Franzis-Verlag.

Dr. Helmut Weber

MS-DOS mit Komfort

Mehrere Programme in einer Kommandozeile aufrufen

Ein Turbo-Pascal-Programm ruft die in der Kommandozeile übergebenen Programme bzw. MS-DOS-Kommandos nacheinander auf. Ein kleiner Leckerbissen in diesem Programm ist die Prozedur zum Suchen des DOS-Environments.

Für MS-DOS-Anwender ist es auf die Dauer ermüdend, daß DOS nur ein Kommando in der Kommandozeile zuläßt. Schön wäre doch eine Aufrufsequenz nach Unix-Art:

PAS1 prog,,con,nul; PAS2; LINK prog ,,,pascal; DEL prog.obj; DIR prog.*
Damit wird ein Pascal-Programm mit MS-Pascal übersetzt und gelinkt während der Programmierer in Ruhe Kaffee trinken kann! Die einzelnen Kommandos wurden durch Semikolons getrennt. Natürlich geht das mit einer Stapel-Datei auch recht gut, aber man scheut doch bei individuellen, nur einmal vorkommenden Kommandosequenzen häufig den Aufwand und möchte nicht mehr eintippen als unbedingt notwendig ist. Um dies zu erreichen, sind keinerlei Eingriffe in MS-DOS nötig. Zur Realisierung reicht Turbo-Pascal aus, wenn die EXEC-Funktion [1] benutzt wird. Das Programm (unser Autor hat es X genannt) bekommt die auszuführende Kommando-Sequenz als Argumentstring mit auf den Weg:

X PAS1 prog,,con,nul; PAS2; LINK prog ,,,pascal; DEL prog.obj; DIR prog.*

Die zusätzliche Eingabe von X wird hoffentlich niemanden stören. Wenn das Semikolon Teil eines Kommandos ist, wie z.B. „PAS1 prog;" oder „LINK prog; ," sollte ein alternatives Trennzeichen verwendet werden. Unser Autor hat dafür das Doppelkreuz (#) reserviert. Sobald das Doppelkreuz in der Kommandozeile auftaucht, wird dieses zum Trennzeichen und das Semikolon wird ungeändert weitergegeben. Tippfaule dürfen damit schreiben:

X PAS1 prog;# PAS2# LINK prog;# DEL prog.obj# DIR prog.*

DOS-EXEC unter Turbo-Pascal

Die Prozedur GETARGS zerlegt den übergebenen String in die einzelnen Kommandos und speichert diese im Feld COMMAND. Die Prozedur FINDENV sucht im DOS-Environment den Wert der Variablen COMSPEC. Diese Systemvariable gibt an, in welchem Directory sich der Kommandoprozessor befindet. EXEC und SETBLOCK sind Prozeduren zum Starten eines beliebigen MS-DOS-Programms von Turbo-Pascal aus. Das Hauptprogramm führt in einer Schleife die übergebenen Kommandos aus, in der jeweils von EXEC eine Version des COMMAND.COM mit der Option /c, gefolgt vom individuellen Kommando gestartet wird. Damit ist gewährleistet, daß die mit der Systemvariablen PATH spezifizierten Zugriffspfade zum Suchen der Objektprogramme benutzt werden. Kommandos wie A: und die Ausführung von Stapel-Dateien sind möglich.
Etwas Zeit kostet das Nachladen von COMMAND.COM, was jedoch bei einer Festplatte kaum auffällt. Der zur Verfügung stehende Speicherplatz wird durch das geladene Turbo-Pascal-Programm und den Kommando-Prozessor etwas geschmälert. Das Programm wurde unter MS-DOS 2.11 und MS-DOS 3.2 auf einem Comodore PC 20-II und dem MCI AT4SLC getestet. Beim Compilieren sollte der minimale und der maximale freie dynamische Speicher auf 200H gesetzt werden.

Literatur:

[1] *Keul Manfred*: DOS-EXEC-Funktion von Turbo-Pascal aus. mc 1987, Heft 2, Seite 50.

Ein Turbo-Pascal-Programm erleichtert die Arbeit mit dem PC

```
program X; { eXecute }

const
   STACKSIZE = $200;
   MAXCOM = 20;

type
   REGISTER = record
         AX,BX,CX,DX,BP,SI,DI,DS,ES,FLAGS : integer
   end;
   NAMESTRING = string[80];
   string128 = string[128];
   COMARRAY = array [1..MAXCOM] of NAMESTRING;

var
   COMNUMBER, i, j : integer;
   COMMAND : COMARRAY;
   REG : REGISTER;
   PATHNAME, CURCOM : NAMESTRING;

procedure SETBLOCK;
begin
   with REG do
   begin
      BX:= Sseg - Cseg + STACKSIZE;
      ES:= Cseg;
      AX:= $4A00;
      MsDos(REG);
      if odd(FLAGS) then
      begin
         writeln ('SETBLOCK error');
         halt
      end;
   end
end; (* SETBLOCK *)
```

```
procedure EXEC (PROGRAMMNAME, COMMANDLINE : NAMESTRING);
type
   PAR_TYPE = record
         ENV_SEG,COMM_OFF,COMM_SEG,D1,D2,D3,D4 : integer
   end;
const
   PAR_BLOCK : PAR_TYPE = (ENV_SEG : 0; COMM_OFF : 0; COMM_SEG : 0;
                           D1 : 0; D2 : 0; D3 : 0; D4 : 0);
   ASCIIZ    : NAMESTRING = '';
   SSSAVE    : integer = 0;
   SPSAVE    : integer = 0;
   AXREG     : integer = 0;
   FLAGSREG  : integer = 0;
begin
   ASCIIZ:= PROGRAMMNAME + chr (0);
   COMMANDLINE:= COMMANDLINE + chr (13);
   with PAR_BLOCK do
   begin
      COMM_SEG:= seg (COMMANDLINE);
      COMM_OFF:= ofs (COMMANDLINE);
   end;
   inline(              { Save registers DS, BP, SP, SS }
         $1E/             { PUSH DS }
         $55/             { PUSH BP }
         $2E/             { CS:     }
         $89/$26/SPSAVE/  { MOV [SPSAVE],SP }
         $8C/$D0/         { MOV AX,SS }
         $2E/             { CS:     }
         $A3/SSSAVE/      { MOV [SSSAVE],AX }
                        { Copy CS into DS and ES }
         $0E/             { PUSH CS }
         $0E/             { PUSH CS }
         $1F/             { POP DS  }
         $07/             { POP ES  }
                        { DS:DX must point to ASCIIZ string with
                          filename to be executed }
         $BA/ASCIIZ+1/    { MOV DX,ASCIIZ + 1 }
                        { ES:BX must point to parameter block
                          containing command line etc. }
         $BB/PAR_BLOCK/   { MOV BX,PAR_BLOCK  }
                        { Interrupt 21, function 4B00 }
         $B8/$00/$4B/     { MOV AX,4B00, EXEC function code }
         $CD/$21/         { INT 21, do it }
                        { After execution restore registers }
         $2E/             { CS:     }
         $8B/$1E/SSSAVE/  { MOV AX,[SSSAVE] }
         $8E/$D3/         { MOV SS,BX }
         $2E/             { CS:     }
         $8B/$26/SPSAVE/  { MOV SP,[SPSAVE] }
         $5D/             { POP BP  }
         $1F/             { POP DS  }
                        { Save EXEC error codes in AX and Flags }
         $2E/             { CS:     }
         $A3/AXREG/       { MOV [AXREG],AX  }
         $9C/$58/         { PUSHF, POP AX   }
         $2E/             { CS:     }
         $A3/FLAGSREG     { MOV [FLAGSREG],AX  }
      );
   with REG do
   begin
      AX:= AXREG; FLAGS:= FLAGSREG;
      if odd (FLAGS)  then
      begin
         writeln ('EXEC error!, AX = ', AX);
         halt
      end
   end;
end; { EXEC }

procedure FINDENV (sysvar : string128; var env : NAMESTRING);
var find : string128;
    found : boolean;
    pint : ^integer;
    pfind : ^char;
    i, seg, ofs : integer;

 procedure NEW_PFIND;
 begin
    ofs:= ofs + 1; pfind:= ptr (seg, ofs)
 end; { NEW_PFIND }

begin
   found:= false;
   for i:= 1 to ord (sysvar[0]) do sysvar[i]:= upcase (sysvar[i]);
   pint:= ptr (cseg, $002C);
   seg:= pint^; ofs:= 0;
   env:= '';
   pfind:= ptr (seg, ofs);
   while (not found) and (ofs >= 0) and (pfind^ <> chr (0)) do
   begin
      find:= '';
      while (pfind^ <> '=') and (pfind^ <> chr (0) ) do
      begin
         find:= find + pfind^;
         NEW_PFIND;
      end;
      if pfind^ = '=' then NEW_PFIND;
      if (sysvar = find) then
      begin
         while pfind^ <> chr (0) do
         begin
            env:= env + pfind^;
            NEW_PFIND;
         end;
         found:= true;
      end else
      begin
         while pfind^<> chr (0) do NEW_PFIND;
         NEW_PFIND;
      end;
   end;
end; { FINDENV }

procedure GETARGS (var no : integer; var com : COMARRAY);
const
    firstsep = ';';
    secondsep = '#';
var i, m : integer;
    separator : char;
    s : NAMESTRING;
    line:string[128] absolute cseg:$0080;
begin
   if length (line) = 0 then halt;
   i:= pos (secondsep, line);
   if i = 0 then separator:= firstsep
   else separator:= secondsep;
   no:= 0;
   m:= pos (separator, line);
   while m <> 0 do
   begin
      if m = 1 then
         if length (line) = 1 then line:= ''
         else line:= copy (line, 2, length (line) -1 )
      else
      begin
         no:= succ (no);
         if no > MAXCOM then exit;
         com[no] := copy (line, 1, m-1);
         line:= copy (line, m+1, length (line) - m);
      end;
      m:= pos (separator, line);
   end;
   if length (line) > 0 then
   begin
      no:= succ (no);
      if no > MAXCOM then exit;
      com[no]:= line
   end;
end; { GETARGS }

begin  { X }
   FINDENV ('COMSPEC', PATHNAME);
   GETARGS (COMNUMBER, COMMAND);
   SETBLOCK;
   writeln;
   for i:= 1 to COMNUMBER do
   begin
      CURCOM:= COMMAND[i];
      j:= 1;
      while CURCOM[j] = ' ' do
      begin
         CURCOM:= copy (CURCOM, 2, ord (CURCOM[0]) - 1);
         j:= succ (j)
      end;
      writeln; writeln (CURCOM);
      EXEC (PATHNAME, '/c ' + CURCOM);
   end;
end. { X }
```

Arne Schirmacher

Assembler-programm für den Macintosh

Das Assemblerprogramm dieses Artikels, der Screen Saver, installiert eine ebenso simple wie praktische Zusatzfunktion im Apple Macintosh Computer. Es belegt etwas mehr als 300 Bytes im RAM und etwa 400 Bytes auf der Diskette. Sobald eine einstellbare Zeit lang weder eine Taste betätigt noch der Mausknopf gedrückt wurde, wird der Bildschirminhalt gelöscht. Der Screen Saver wartet, bis der Benutzer eine Taste betätigt; dann wird der Bildschirminhalt blitzschnell rekonstruiert.

Wie es sich für ein gutes Macintosh Programm gehört, arbeitet es völlig idiotensicher. Einmal in der Systemdatei installiert, ist es praktisch unsichtbar, startet bei jedem Booten des Computers automatisch und wird bei jedem Kopieren der Systemdatei auf eine neue Diskette gleich mitkopiert.

Nun fragen sich vielleicht einige Leser, warum man denn den Macintosh nicht einfach ausschaltet, wenn man mit den Arbeiten aufhören möchte. Der Grund liegt darin, daß der Anwender sehr häufig z. B. eine RAMDisk mit verschiedenen, häufig benötigten Dokumenten angelegt hat. Oder er hat eine Festplatte, die ja auch eine gewisse Zeit zum Hochfahren benötigt. Manche Programme brauchen auch bis zu einer Minute, bis sie endlich von der Diskette geladen sind.

Auf der anderen Seite schadet es der Bildröhre, wenn der Computer nicht ausgeschaltet wird und Tag und Nacht das überwiegend weiße Bildschirmfenster zu sehen ist. Das ist jetzt kein Witz; bei meinem Macintosh kann ich bei dunkler Bildröhre schon schwach die Umrisse der Menüleiste erkennen, die sich in den Leuchtphosphor eingebrannt hat.

Also verwendet man den Screen Saver, macht sich keine Sorgen um den Bildschirm und hat trotzdem die Computerleistung blitzartig zur Verfügung. Es kostet einen nur ein paar Bytes RAM. Aber das merkt man sowieso nicht.

Das Assemblerprogramm wurde mit dem Macroassembler MacASM von Mainstay entwickelt. Mir persönlich gefällt er von den zur Zeit erhältlichen Assemblern am besten, denn er arbeitet extrem schnell. Wer einen anderen Assembler verwendet, muß dann die MacASM – spezifischen Befehle entsprechend modifizieren. Der Code an sich paßt natürlich zu jedem 68000 Assembler, da dürfte es keine Schwierigkeiten geben.

Wie der Screen Saver arbeitet

Das Listing des Screen Savers (*Bild*) besteht aus vier Teilen: dem Initialisierungsteil, aus programmtechnischen Gründen am Ende des Listings; einer Routine zur Überwachung der Tastatur; einem Unterprogramm zum Löschen und Wiederherstellen der Bildschirmfenster und einem Unterprogramm, das auf einen Tastendruck wartet und in der Wartezeit für ein wenig Bewegung auf dem Bildschirm sorgt. Der Benutzer muß ja merken, daß der Computer in Wirklichkeit noch an ist, sonst betätigt er womöglich den Netzschalter – mit fatalen Folgen.

Wie schon erwähnt, wird der Screen Saver während der Initialisierungsphase geladen und gestartet. Man kann ihn ungefähr bei Adresse $2D00 im RAM finden. Die erste Anweisung ist eine Verzweigung ans Ende des Programms zum Initialisierungsteil. Dieser Teil liegt ganz am Ende des Blocks, weil er nur ein einziges Mal gebraucht und nach Beendigung gelöscht wird.

Im Macintosh werden durch den Videoschaltkreis jede Sechzigstelsekunde In-

```
00140 * Screen Saver installiert sich selbst im Betriebssystem des Macintosh
00150 * und loescht den Bildschirm, falls mindestens 30 Minuten lang keine
00160 * Taste oder der Mausknopf betaetigt wurde. Sobald waehrend des
00170 * Standbys eine Taste betaetigt wird, stellt Screen Saver den
00180 * urspruenglichen Bildschirminhalt wieder her.
00190 *
00200 *--------------------------------
00210                   LIST OFF
00220                   INCLUDE Library.Asm
00230                   LIST ON
00240 *--------------------------------
00250 ;
00260 Ticks             EQU  $16A       Ticks (1/60 Sek.) seit Systemstart
00270 MBTicks           EQU  $16E       Zeitpunkt des letzten Knopfdrucks
00280 KeyTime           EQU  $186             "     des letzten Tastendrucks
00290 SysZone           EQU  $2A6       Ptr auf System Heap
00300 ScrnBase          EQU  $824       Ptr auf VideoRAM-Basisadresse
00310 CurrentA5         EQU  $904       korrekter Wert des Registers A5
00320 GrayRgn           EQU  $9EE       Handle (Ptr auf Ptr) der grauen
00330 ;                                 Schreibtischflaeche
00340 DMtrOff           EQU  $DFF1FF    Adresse Disk Motor stoppen
00350 ;
00360 evtBlkSize        EQU  16         Laenge des event - Records
00370 what              EQU  0          Offset zu event Art
00380 when              EQU  6          Offset zu event Zeitpunkt
00390 ;
00400 ;                                 Offsets in den Queues
00410 qType             EQU  4          Queue - Typ
00420 vblAddr           EQU  6          Adresse der aufzurufenden Routine
00430 vblCount          EQU  10         Verzoegerung in Ticks
00440 vblPhase          EQU  12         Verschiebung zu Tasks gleicher Frequenz
00450 ;
00460 NullEvent         EQU  0          Konstante fuer "Kein Event"
```

Wer den Bildschirm seines Apple-Masintosh schonen will, braucht dieses Programm

```
00470 screenBitsRect   EQU  -116       Offset zu Konstante in GrafGlobals
00480 ;
00490 *---------------------------------
00500                  TFILE "Buffer.Bin"
00510                  RFILE "Screen Saver 1.4",????,????,0
00520 *---------------------------------
00530 * Dieses Programm ist eine INIT - Resource. Nach der Assemblierung
00540 * muss die Resource mit RMover, ResEdit oder einem aehnlichen Programm
00550 * in die SYSTEM - Datei kopiert werden. Die ID kann dabei in den Grenzen
00560 * von 1 - 32 geaendert werden, um Ueberschneidungen mit schon vorhandenen
00570 * INIT's zu vermeiden.
00580 * Bei Systemstart laedt das Betriebssystem nacheinander die INIT's mit
00590 * der ID 1 - 32 und startet sie.
00600 *
00610                  RSRC INIT,20,80,"Screen Saver"
00620 ;
00630 First    BRA.L   Install                 hier ist der Einsprung in
00640 ;                                        die INIT - Resource
00650          ASC     'ARNE'                  um Code im RAM zu identifizieren
00660 vblRoutine
00670 *
00680 * Diese Routine wird vom Betriebssystem einmal pro Sekunde gestartet,
00690 * nachdem sie von Install in die Kette der vertical retrace interrupts
00700 * eingefuegt wurde. Sie ueberprueft, ob der letzte Tasten- oder Maus-
00710 * knopfdruck schon mindestens 30 Minuten (ggf. auch eine andere Zeit
00720 * je nach Inhalt von DelayTicks) zurueckliegt und startet eventuell
00730 * die eigentliche Standby - Routine BlankScreen.
00740 *
00750          MOVEM.L D0/A0,-(SP)             Register retten
00760          MOVE.L  Ticks,D0                Uhrzeit in Register D0
00770          SUB.L   MBTicks,D0              wurde innerhalb des Limits
00780          CMP.L   DelayTicks(PC),D0       (30 min.) Knopf betaetigt?
00790          BLS.S   .0                      ja -> abbrechen
00800          MOVE.L  Ticks,D0                nein -> Zeitpunkt des letzten
00810          SUB.L   KeyTime,D0              Tastendrucks genauso
00820          CMP.L   DelayTicks(PC),D0       checken
00830          BLS.S   .0                      falls keine Vorkommnisse in den
00840          BSR.S   BlankScreen             letzten 30 min --> Standby
00850 .0       LEA     vblElement+vblCount(PC),A0      Adresse Taskzaehler
00860          MOVE.W  #60,(A0)                wieder mit 60 Ticks laden
00870          MOVEM.L (SP)+,D0/A0             Register wiederherstellen
00880          RTS                             und vblTask verlassen
00890 ;
00900 BlankScreen
00910 *
00920 * BlankScreen besorgt das Loeschen und Wiederherstellen des Bildschirm-
00930 * inhaltes. Das Bitmuster des Menubalkens wird auf den Stack gerettet,
00940 * weil manche Programme auf ein Menu verzichten und man in dem Falle
00950 * die Menu Manager Prozedur DrawMenuBar nicht verwenden kann.
00960 * Die Wiederherstellung des Schreibtischhintergrundes und der Fenster
00970 * werden durch Window Manager Prozeduren erreicht, die ihrerseits
00980 * update events generieren, die dann die Programme veranlassen (sollten),
00990 * den Fensterinhalt aufzufrischen.
01000 *
01010          MOVEM.L D0-D7/A0-A6,-(SP)       und wieder Register retten
01020          TST.B   DMtrOff                 Diskette abschalten
01030          MOVE.L  A5,D0                   A5 ueberpruefen
01040          ANDI.L  #$FFFFFFFE,D0           Bit 0 immer auf 0, denn
01050          MOVE.L  D0,A5                   vielleicht ist A5 ungerade
01060          MOVE.L  screenBitsRect(A5),D0   ist A5 gueltiger Ptr auf
01070          CMPI.L  #$01560200,D0           QuickDraw Globals? (1 Konstante
01080          BEQ.S   .0                      ueberpruefen)
01090          MOVE.L  CurrentA5,A5            nein --> A5 korrigieren
01100 .0       TBX     HideCursor              Cursor verschwinden lassen
01110          MOVE.W  #20*16-1,D0             Menubar ist 20 Pixel hoch
01120          MOVE.L  ScrnBase,A0             linke obere Bildschirmecke
01130 .1       MOVE.L  (A0),-(SP)              Menubar auf Stack kopieren
01140          ADDQ.L  #4,A0                   naechstes Langwort
01150          DBRA    D0,.1                   D0 erniedrigen, Exit falls neg.
01160          MOVE.W  #342*16-1,D0            Schleifenzaehler fuer
01170          MOVE.L  ScrnBase,A0             das Loeschen des VideoRAM's
01180 .2       MOVE.L  #$FFFFFFFF,(A0)         loeschen
01190          ADDQ.L  #4,A0                   naechste Bildschirmadresse
01200          DBRA    D0,.2                   Loop durchfuehren
01210          BSR.S   MoveApple               den Apfel laufen lassen
01220          CLR.L   -(SP)                   graue Schreibtischoberflaeche
01230          MOVE.L  GrayRgn,-(SP)           wiederherstellen
01240          TBX     PaintOne
01250          SUBQ.L  #4,SP                   Ptr auf oberstes Fenster
01260          TBX     FrontWindow             holen
01270          MOVE.L  GrayRgn,-(SP)           dieses Fenster und alle darunter
01280          TBX     PaintBehind             liegenden zeichnen und updaten
01290          MOVE.W  #20*16-1,D0             Menubar wiederherstellen
01300          MOVE.L  A7,A1                   (Man kann nicht "DrawMenuBar"
01310          ADD.L   #64*20,A1               verwenden, weil manche Programme
01320          MOVE.L  ScrnBase,A0             den MenuManager nicht initialis.
01330 .3       MOVE.L  -(A1),(A0)              und es dann zum Systemcrash
01340          ADDQ.L  #4,A0                   kommt)
01350          DBRA    D0,.3
01360          ADD.L   #64*20,A7               StackPtr korrigieren
01370          TBX     ShowCursor              Cursor wieder zeigen
```

terrupts erzeugt. Eine Reihe von Prozessen sind von diesen Interrupts abhängig; aus diesem Grund existiert der sogenannte „Vertical Retrace Manager" (engl. für Zeilenrücklauf), der alle diese Routinen koordiniert. Der Initialisierungsteil tut nichts weiter, als eine weitere Routine zu den schon vorhandenen hinzuzufügen. Wir können uns darauf verlassen, daß ab sofort unsere Routine pünktlich zum gewünschten Zeitpunkt, der in Sechzigstelsekunden (Ticks) angegeben wird, startet.

Nach genau einer Sekunde tritt dieser Fall erstmals für den Screen Saver ein. Die Routine vblRoutine, die nun läuft, muß entscheiden, ob der Macintosh schon lange genug nicht mehr benutzt wurde und dann gegebenenfalls das Unterprogramm BlankScreen aufrufen, das dann für die Verdunkelung des Bildschirms sorgt.

Das geht sehr einfach, da in zwei Systemvariablen jeweils die Uhrzeiten (in Ticks seit Systemstart) der letzten Maus- und Tastaturbetätigung enthalten sind. In einer weiteren Systemvariablen ist die aktuelle Zeit enthalten. Es ist also lediglich die Differenz zwischen der jetzigen Zeit und dem Zeitpunkt der letzten Tastenbetätigung zu berechnen und mit einem Maximalwert, enthalten in der Konstanten DelayTicks, zu vergleichen. Falls dieser Zeitraum überschritten wurde, erfolgt ein Sprung zum Unterprogramm BlankScreen, sonst nicht.

Die letzte Tätigkeit, die vblRoutine durchführt, ist, den Zähler für die Verzögerung wieder mit einem neuen Wert zu laden. Falls man das nicht macht, würde diese Routine nicht wieder durch einen Interrupt gestartet werden können, denn unmittelbar vor Start der Interruptroutine wird dieser Zähler auf Null gesetzt, also deaktiviert.

Das Unterprogramm BlankScreen besorgt, wie gesagt, das Löschen und Rekonstruieren des Bildschirms. Der Code mag vielleicht etwas seltsam aussehen, aber es funktioniert leider nicht anders, aus zwei Gründen:

1. Der Screen Saver wird durch einen Interrupt gestartet. Kein Register des Prozessors wird einen definierten Wert haben. Das Adreßregister A5 dient meistens als Pointer auf bestimmte Systemvariablen, hat oft also den richtigen Wert, aber z. B. verwenden Microsoft-Programme dieses Register für ihre eigenen Zwecke. Ist A5

```
01380           MOVEM.L (SP)+,D0-D7/A0-A6       Register wiederherstellen
01390           RTS                             und Rueckkehr
01400 ;
01410 MoveApple
01420 *
01430 * MoveApple ueberwacht die Tastatur und laesst, solange sich da nichts
01440 * tut, einen Apfel ueber den Bildschirm wandern.
01450 *
01460           LINK    A6,#-evtBlkSize         Platz fuer lokale Variable
01470 .0        MOVEQ   #20,D1                  21*16 Zeilen
01480           MOVE.L  ScrnBase,A1             A1 ist Adresse des Apfelbildes
01490 .1        MOVEQ   #31,D2                  32 Spalten
01500 .2        MOVEQ   #10,D3                  Apfelbild ist 11 Zeilen hoch
01510           MOVE.L  A1,-(SP)                Adresse fuer spaeter retten
01520           LEA     Apple(PC),A0            Adresse des Bitmusters in A0
01530 .3        MOVE.W  (A0)+,D0                naechste Apfelzeile holen
01540           EOR.W   D0,(A1)                 mit Bildschirm verknuepfen
01550           ADDA.L  #64,A1                  naechste Zeile
01560           DBRA    D3,.3                   weiter bis fertig
01570           MOVE.L  (SP)+,A1                Adresse des Apfels wiederherst.
01580           MOVEQ   #60,D4                  60 Ticks = 1 Sekunde
01590           ADD.L   Ticks,D4                D4 enthaelt Zeit des n. Durchlfs
01600 .4        LEA     -evtBlkSize(A6),A0      Register fuer OSEvent vorber.
01610           MOVEQ   #-1,D0
01620           TBX     GetOSEvent              Tastendruck?
01630           BEQ.S   .6                      ja -- zurueck zu BlankScreen
01640           CMP.L   Ticks,D4                nein, ist die Sek. schon um?
01650           BHI.S   .4                      nein, warten
01660           MOVE.W  #10,D3                  Apfel wieder wegmachen
01670           LEA     Apple(PC),A0
01680           MOVE.L  A1,-(SP)
01690 .5        MOVE.W  (A0)+,D0
01700           EOR.W   D0,(A1)
01710           ADDA.L  #64,A1
01720           DBRA    D3,.5
01730           MOVE.L  (SP)+,A1                aus alter Position die
01740           ADDQ.L  #2,A1                   neue Position berechnen
01750           DBRA    D2,.2                   un einen Schritt weiter
01760           ADDA.L  #15*64,A1               eine Zeile weiter runter gehen
01770           DBRA    D1,.1
01780           BRA.S   .0
01790 .6        UNLK    A6                      lokale Var's vernichten
01800           RTS                             Rueckkehr zu BlankScreen
01810 ;
01820 Apple     DATA    /%0000000011000000      Bitmuster des kleinen Apfels
01830           DATA    /%0000000110000000
01840           DATA    /%0000000100000000
01850           DATA    /%0000111011100000
01860           DATA    /%0001111111110000
01870           DATA    /%0001111111000000
01880           DATA    /%0001111111000000
01890           DATA    /%0001111111110000
01900           DATA    /%0001111111110000
01910           DATA    /%0000111111100000
01920           DATA    /%0000011011000000
01930 ;
01940 vblElement        DEFS 14                 14 Bytes Platzbedarf
01950 DelayTicks        DATA 30*60*60           Zeitkonstante: 30 Minuten
01960 Last
01970 Install
01980 ;
01990 *---------------------------------
02000 *
02010 * Bei Systemstart wird dieser Programmteil ein einziges Mal ausgefuehrt.
02020 * Er installiert die vblRoutine in der Interruptqueue des Macintosh.
02030 * Anschliessend wird der Initialisierungsteil von der Resource getrennt,
02040 * sodass der Speicher vom Memory Manager spaeter wieder verwendet
02050 * werden kann.
02060 *
02070           LEA     vblElement(PC),A0       Adresse des neuen Eintrags in
02080 ;                                         die vertical retrace queue
02090           MOVE.W  #1,qType(A0)            1 ist vblType
02100           LEA     vblRoutine(PC),A1       Adresse der Routine berechnen
02110           MOVE.L  A1,vblAddr(A0)          und eintragen
02120           MOVE.W  #60,vblCount(A0)        Routine soll 1mal pro Sek. lauf.
02130           MOVE.W  #9,vblPhase(A0)         Verschiebung - Wert ist egal
02140           TBX     VInstall                Task in Interruptkette einfuegen
02150           LEA     First-4(PC),A1          enthaelt Offset des Master-
02160           MOVE.L  (A1),A0                 ptrs vom Heapbeginn
02170           ADD.L   SysZone,A0              Heapstart dazuaddieren
02180           MOVE.L  A0,A2                   A0 ist jetzt Handle auf Res.
02190           TBX     HUnlock                 entriegeln
02200           MOVE.L  A2,A0
02210           MOVE.L  #Last-First,D0          den Initialisierungsteil
02220           TBX     SetHandleSize           herausnehmen
02230           MOVE.L  A2,A0                   und den Rest wieder verriegeln
02240           TBX     HLock
02250           MOVE.L  A2,D7                   fuer das Betriebssystem erf.
02260           RTS
02270 ;
02280           ENDR
02290 ;
02300           END
```

aber nicht korrekt, funktionieren die Betriebssystemaufrufe nicht. Laut Programmierhandbuch „Inside Macintosh" bekommt man den korrekten Wert für A5 aus einer Systemvariablen, nämlich aus CurrentA5. Stimmt bloß nicht immer, wie die Praxis zeigt. Ausgerechnet beim MiniFinder, einem Apple-Programm, funktioniert das nicht. Daher muß man zunächst einmal checken, ob A5 einen plausiblen Wert hat. Ist das nicht der Fall, müssen wir notgedrungen den Wert aus CurrentA5 übernehmen, auf den man sich leider nicht unbedingt verlassen kann. Bis jetzt hat es aber immer funktioniert.

2. Ein weiteres Problem ist die Rekonstruktion der Menüzeile. Man könnte eine schöne Prozedur aufrufen, die die Menüzeile druckt. Wenn aber das laufende Programm den Menü-Manager nicht initialisiert hat, weil es ohnehin keine Menüs verwendet, kommt es zu Systemcrash. Das passiert auch wieder beim Minifinder. Daher müssen wir die Menüzeile irgendwohin retten, um sie später wieder zurückzukopieren (Zeilen 1060–1100). Der beste Platz dafür ist der Stack. Auch hier könnten Schwierigkeiten auftreten, denn die 1,2 KByte für die Menüzeile sind ein wenig außerhalb des garantierten Fassungsvermögens des Stacks von 1 KByte.

Mit einer simplen Schleife, die lauter $FFFFs ins VideoRAM schreibt, wird der Bildschirm schwarz. Dann folgt die Routine MoveApple, danach kommt wieder ein interessantes Stück, nämlich die Regenerierung des Bildschirms. Zunächst zeichnet die Window Manager Prozedur PaintOne den grauen Schreibtischhintergrund. Mit der Funktion FrontWindow wird ein Pointer auf das vorderste Fenster ermittelt und gleich der Prozedur PaintBehind übergeben. Diese Prozedur generiert die update events für sämtliche vorhandenen Fenster. Ein update event ist ein Befehl des Betriebssystems an das laufende Programm, ein bestimmtes Fenster neu zu zeichnen. Diese events werden aber erst von der Applikation bearbeitet, nachdem der Prozessor mit dem Screen Saver fertig und wieder in die Hauptschleife der Applikation zurückgekehrt ist. Nun wird die Menüzeile wiederhergestellt, indem das Bitmuster vom Stack in die obersten zwanzig Zeilen des VideoRAMs kopiert wird. Dann noch den Cursor wieder anstellen und die Register mit den alten Werten laden – das war's.

Manche Programme sind so konzipiert, daß eigentlich niemals ein update event auftreten könnte. Das sind alle Programme, die nur ein Fenster haben und kein Schreibtischzubehör unterstützen. Folglich wird zwar der Fensterrahmen gezeichnet, der Inhalt bleibt aber leer oder unvollständig.

Vorsichtshalber legt die Routine BlankScreen die Diskette still (Zeile 970), falls sie noch laufen sollte. Das führt garantiert zum späteren Absturz, aber meiner Ansicht nach ist das besser, als wenn irgendwann durch einen unglücklichen Zufall stundenlang das Laufwerk leer läuft und vielleicht die Diskette zerstört. Normalerweise steht das Laufwerk, wenn der Screen Saver zum Einsatz kommt, dann passiert natürlich auch nichts. Aber diesen Befehl kann man auch weglassen.

MoveApple tut nichts weiter, als die Tastatur zu überwachen und in der Wartezeit ein kleines Apfelbildchen über dem Bildschirm zu bewegen.

Die folgenden Zeilen kopieren den kleinen Apfel auf den Bildschirm. Leider dürfen wir hier nicht QuickDraw-Routinen einsetzen, aus folgendem Grunde: Wenn QuickDraw auf den gesamten Bildschirm zeichnen soll und nicht nur in den Grenzen des (jetzt unsichtbaren) vordersten Bildschirmfensters, dann müßten wir eigens einen grafPort dafür eröffnen. Ein grafPort ist so etwas wie ein Fenster auf Betriebssystemebene. Die QuickDraw-Funktion OpenPort, die einen neuen grafPort eröffnet, benutzt einige der Memory Manager Prozeduren, um Speicherplatz für Daten zu belegen.

Dabei könnte folgende Situation auftreten: Bevor der Interrupt ausgelöst wurde, der den Screen Saver startete, war der Memory Manager, der für die Speicherverwaltung zuständig ist, gerade bei der Arbeit und hat z. B. verschiedene Pointer geändert. Nun läuft der Screen Saver und benutzt dabei über Umwege den Memory Manager. Der Speicher ist aber in Unordnung, weil ja ein paar Millisekunden vorher der Memory Manager daran gehindert wurde, seine Arbeit ordnungsgemäß zu beenden. Es kommt dann zum Systemcrash. Zwar ist es unwahrscheinlich, daß nach einer Wartezeit von 30 Minuten das Programm nicht in einer Warteschleife hängt, aber es könnte ja sein.

Also befördern wir das Bildchen eigenhändig ins VideoRAM. Das tun die Zeilen 1420–1510. Die Systemvariable Ticks enthält die Uhrzeit und wird jede Sechzigstelsekunde durch den Interrupt (nicht durch einen vblTask – die sind gesperrt, solange ein anderer vblTask ausgeführt wird) heraufgezählt. So kann der Programmierer schöne Warteschleifen realisieren. In der Warteschleife dieses Programmteils wird aber noch mit der Funktion GetOSEvent die Tastatur gelesen. Falls da eine Taste betätigt wurde, möchte der Anwender weiterarbeiten und diese Routine ist beendet, wenn nicht, löscht die Printschleife den Apfel, indem sie ihn nochmal druckt (durch die EOR-Verknüpfung verschwindet er dann) und stellt die Druckposition einen Schritt weiter. Dann geht alles von vorne los.

Resourcen in der Systemdatei

Was ist eigentlich in der Systemdatei drin, hat sich sicher schon mancher gefragt. Macintosh-Dateien bestehen im allgemeinen nicht aus einem großen Block an Daten, die sequentiell gelesen oder geschrieben werden. Sie enthalten eine Reihe von sogenannten Resourcen, geordnet nach Resourcetypen. Zum Beispiel sind alle Zeichensätze in Resourcen gespeichert, ebenso Tabellen für Fenster und Dialoge, Bildchen, Schreibtischzubehör, Treiber für Peripheriegeräte und noch einiges mehr.

Der Resourcetyp „INIT" kommt nur in der Systemdatei vor. Während der Startphase lädt das Betriebssystem alle Resourcen vom Typ „INIT" in der Nummer 1–32 (soweit vorhanden), die stets ein Maschinenprogramm enthalten, und startet sie. Die darin enthaltenen Programme installieren z. B. Patches für fehlerhafte ROM-Routinen, eine spezifische Tastaturbelegung oder in unserem Falle eben den Screen Saver.

Es gibt inzwischen einige Resource-Editoren, mit deren Hilfe jeder diese Resourcen löschen, kopieren oder modifizieren kann, um etwa englische Anweisungen (die immer in Resourcen abgelegt sind) ins Deutsche zu übersetzen. Es lohnt sich, einmal verschiedene Dateien und Programme mit einem solchen Editor zu durchstöbern.

Assemblieren mit MacASM und Installation

Man tippt das abgedruckte Listing fehlerlos ein, sichert es mit save Screen Saver.Asm oder so und assembliert es durch asm. Der Assembler produziert nun eine Binärdatei mit dem assemblierten Code. Diese Datei muß nun noch durch den Resource Compiler laufen. Dazu tippt man den Befehl dos ein und klickt vom Finder aus auf das Abbild mit dem Namen „Buffer Bin". Jetzt startet nicht der Assembler, sondern der integrierte Resource Compiler, der aus dieser Datei die INIT-Resource herstellt. Nach ein paar Sekunden ist man wieder im Finder, auf dem Schreibtisch gibt es nun ein weiteres Dokument mit dem Namen „Screen Saver 1.2". Falls jemand sein Listing unter diesem Namen abgespeichert hatte, ist es jetzt weg.

Dieses Dokument enthält unsere INIT-Resource. Die muß nun in die Systemdatei hinein, damit der Screen Saver bei Systemstart geladen wird. Dazu nehmen wir einen Resource-Editor, öffnen das Dokument „Screen Saver 1.2" und selektieren die INIT-Resource mit der Nummer 20 und dem Namen „Screen Saver". „Copy" aus dem Edit-Menü auswählen, die Systemdatei öffnen, auch hier den Init-Typ öffnen oder selektieren (unterschiedlich) und dann „Paste". Jetzt sollte eine Kopie dieser Resource in der Systemdatei vorhanden sein. Falls schon eine INIT-Resource mit der ID-Nummer 20 vorhanden ist, können Sie die ID-Nummer der Screen Savers in eine noch nicht verwendete ändern. Sie ist für die Funktion nicht wichtig.

Das war der Kopiervorgang. Zum Finder zurückkehren, Disketten auswerfen lassen und den Computer aus- und wieder einschalten. Jetzt Diskette wieder einlegen und booten lassen. Nun ist der Screen Saver aktiv. Diese Systemdatei kann nun auf jede andere Diskette kopiert werden, die ebenfalls als Startdiskette benutzt wird.

Sie sollten aber unbedingt wenigstens ein System haben, das keinen Screen Saver enthält. Wenn einmal irgendwelche extrem langen Kalkulationen durchzuführen sind oder längere Texte gedruckt werden sollen, darf der Macintosh keinesfalls mittendrin stillgelegt werden. Natürlich kann man auch die Konstante DelayTicks beliebig erhöhen. Oder vielleicht schreibt sich jemand z. B. ein neues Schreibtischzubehör, das vorübergehend den Screen Saver außer Gefecht setzt. Es ist alles machbar.

Wenn Sie keinen Resource-Editor haben, hat ihn bestimmt ein anderer Macintosh-User in Ihrer Nähe. Diese Editoren sind so nützlich, daß eigentlich jeder Anwender einen haben sollte.

Herwig Feichtinger

GW-Basic ruft MS-DOS

Zwar ist der Befehlssatz von IBM- bzw. GW-Basic recht komfortabel, doch auf manche Möglichkeiten des Betriebssystems hat man keinen unmittelbaren Zugriff. Eine kurze Maschinenroutine erlaubt jedoch den direkten Aufruf der Betriebssystem-Funktionen über den MS-DOS-Aufruf INT 21H.

Was man mit dem MS-DOS-Interrupt 21H alles anstellen kann, wurde in mc schon vor einiger Zeit in der Serie „MS-DOS intern" ausführlich dargestellt. Wer allerdings Programme in Basic schreibt, hat normalerweise nur auf solche Funktionen Zugriff, für die der Interpreter bzw. Compiler auch Befehle zur Verfügung stellt. Das Einlesen der Dateinamen eines Directory in ein String-Array, das Feststellen des Datums oder der Länge eines Directory-Eintrags und vieles andere mehr, läßt sich deshalb so zunächst nicht realisieren.

Abhilfe schafft ein kurzes Maschinenprogramm (*Bild 1*), dem das aufrufende Basic-Programm die typischerweise benötigten CPU-Registerinhalte übergeben kann und das diese anschließend (ggf. verändert) wieder zurückliefert. Ferner wird das Carry-Flag zurückgegeben, das bei vielen MS-DOS-Funktionen einen Fehlerfall signalisiert.

Die Parameterübergabe erfolgt über den Stack, wobei vorausgesetzt wird, daß es sich bei allen Parametern um 2-Byte-Integer-Variablen handelt. Das Programm ist vollständig relokatibel, also nicht an eine bestimmte Anfangsadresse gebunden.

Wenn man den Basic-Interpreter verwendet, kann man das gesamte Maschinenprogramm bequem in ein Integer-Array packen, wie *Bild 2* zeigt. Dieses Beispielprogramm tut so etwas ähnliches wie der Basic-Befehl FILES, stellt also das Inhaltsverzeichnis einer Diskette auf dem Bildschirm dar. Subdirectories werden hierbei mit einem nachgestellten

```
100 REM --- MS-DOS-FUNKTIONSAUFRUFE VON GWBASIC AUS (BEISPIEL) ---
110 REM --- Assembler-UP in Array PRG% laden (nur für Interpreter!) ---
120 DEF SEG:DIM PRG%(29):FOR I%=0 TO 29:READ E$,I$
130 A=VAL("&H"+E$)+256*VAL("&H"+I$)
140 IF A>32767 THEN PRG%(I%)=A-65536! ELSE PRG%(I%)=A
150 NEXT:AX%=0:BX%=0:CX%=0:DX%=0:CF%=0:REM Vor erstem CALL alle Var.dekl.!
160 '
170 REM --- Directory anzeigen ---
180 LINE INPUT"Directory-Maske: ";E$
190 IF E$=""OR RIGHT$(E$,1)="\"OR RIGHT$(E$,1)=":"THEN E$=E$+"*.*"
200 E$=E$+CHR$(0):DTA$=STRING$(43,0):GOSUB 320:REM Puffer=DTA$ setzen
210 A=VARPTR(E$):GOSUB 350:AX%=&H4E00:CX%=17:GOSUB 290:IF CF% THEN 260
220 I$=MID$(DTA$,31):I$=LEFT$(I$,INSTR(I$,CHR$(0))-1)
230 I%=ASC(MID$(DTA$,22,1)):IF (I% AND 16)>1 THEN I$=I$+"\"
240 PRINT I$,:REM Name anzeigen
250 GOSUB 320:AX%=&H4F00:GOSUB 290:IF NOT CF% THEN 220
260 PRINT:PRINT"Fertig!":END
270 '
280 REM --- UP: DOS-Aufruf (INT 21H) ---
290 A=VARPTR(PRG%(0)):CALL A(CF%,AX%,BX%,CX%,DX%):RETURN
300 '
310 REM --- UP: DTA-Adresse auf DTA$ setzen ---
320 A=VARPTR(DTA$):GOSUB 350:AX%=&H1A00:GOTO 290:REM Funktion 1AH
330 '
340 REM --- UP: Adr.des Strings mit A=Stringholder nach DX% ---
350 A=PEEK(A+1)+256*PEEK(A+2):REM Für Compiler +2,+3 statt +1,+2!
360 IF A>32767 THEN DX%=A-65536! ELSE DX%=A
370 RETURN
380 '
390 REM --- Maschinenprogramm für DOS-Aufruf ---
400 DATA 55,89,E5,8B,76,06,8B,14,8B,76,08,8B,0C,8B,76,0A
410 DATA 8B,1C,8B,76,0C,8B,04,CD,21,8B,76,0E,C7,04,00,00
420 DATA 73,02,FF,0C,8B,76,0C,89,04,8B,76,0A,89,1C,8B,76
430 DATA 08,89,0C,8B,76,06,89,14,5D,CA,0A,00
```

Bild 2. Ein Basic-Beispielprogramm (Interpreter-Version) zur Anzeige eines Disketten-Inhaltsverzeichnisses

```
Aufruf von MS-DOS-Funktionen (INT 21H) von GW-Basic aus
Format: CALL adr(CF%,AX%,BX%,CX%,DX%)
Zurückgegebene Parameter: Register; CF%=Carry

0100 55          PUSH BP                    BP retten
0101 89E5        MOV BP,SP                  BP=Zeiger auf Parameter
0103 8B7606      MOV SI,[BP+06]             Adresse von DX%
0106 8B14        MOV DX,[SI]                Register DX mit DX% laden
0108 8B7608      MOV SI,[BP+08]
010B 8B0C        MOV CX,[SI]                dito CX
010D 8B760A      MOV SI,[BP+0A]
0110 8B1C        MOV BX,[SI]                dito BX
0112 8B760C      MOV SI,[BP+0C]
0115 8B04        MOV AX,[SI]                dito AX
0117 CD21        INT 21                     MS-DOS-Interrupt
0119 8B760E      MOV SI,[BP+0E]             Adresse von CF%
011C C7040000    MOV WORD PTR [SI],0000     CF%=0 annehmen
0120 7302        JNB 0124
0122 FF0C        DEC WORD PTR [SI]          nein, CF%=-1 (=true)
0124 8B760C      MOV SI,[BP+0C]
0127 8904        MOV [SI],AX                AX nach AX% speichern
0129 8B760A      MOV SI,[BP+0A]
012C 891C        MOV [SI],BX                dito BX
012E 8B7608      MOV SI,[BP+08]
0131 890C        MOV [SI],CX                dito CX
0133 8B7606      MOV SI,[BP+06]
0136 8914        MOV [SI],DX                dito DX
0138 5D          POP BP
0139 CA0A00      RETF 000A                  5 Parameter * 2 = 0AH
```

Bild 1. Maschinenprogramm zum Aufruf von MS-DOS-Betriebssystem-Funktionen von GW-Basic aus

„Backslash" gekennzeichnet. Es ist kein Kunststück, statt der Bildschirm-Ausgabe das Abspeichern der Dateinamen in ein String-Array vorzusehen, um die Namen z.B. in einem Menüsystem weiterverwenden zu können.

Der für die Aufnahme der Directory-Informationen nötige Disk-Transfer-Bereich wird hier durch die String-Variable DTA$ dargestellt. Sie wird mit der benötigten Länge vorbelegt, weil es von einem Maschinenprogramm aus nicht zulässig ist, die Länge eines Basic-Strings zu verändern; der Interpreter oder Compiler würde mit seiner Variablen-Verwaltung durcheinanderkommen.

Ein paar Dinge sind bei Änderungen des Programms unbedingt zu beachten. Es muß vermieden werden, daß sich die Adresse von Variablen zwischen der Feststellung ihrer Adresse (Unterprogramm ab 350) und dem Aufruf der Maschinenroutine mit CALL verändert, etwa durch eine Garbage-Collection (also keine Stringoperationen in diesen Programmstücken!) oder durch die Neueinführung bisher nicht verwendeter Variablen (also alle Variablen möglichst vorher auf irgendeinen Dummy-Wert setzen).

Bei Verwendung des Compilers ist die Methode des Abspeicherns eines Maschinenprogramms in einem Integer-Array nicht mehr verwendbar, da der CALL-Befehl sich stets auf ein externes, mit dem Linker dazugebundenes Programm bezieht. Ferner ist für den Compiler eine kleine Änderung in Zeile 350 nötig, weil er zwei (statt ein) Byte für die Stringlänge reserviert, so daß die String-Anfangsadresse ein Byte weiter hinten im Stringholder steht.

Arcus-Sinus in C

Das vorgestellte Modul für den C-Compiler des CP/M-68k berechnet den arcsin(x) des Arguments x. Der zurückgegebene Wert und das Argument sind vom Typ „Float", da dieser C-Compiler kein „Double" kennt bzw. dieses gleich in Float umwandelt.

Mittels einer Intervallschachtelung wird, ausgehend von der Untergrenze u = 0, der Obergrenze o = π/2 und dem Startwert π/4 (= Intervallmitte), sin(a) berechnet und mit dem Argument x verglichen. Bei Gleichheit mit Float-Genauigkeit wird abgebrochen und a als arcsin(x) zurückgegeben. Ist sin(a) kleiner als x, so wird a zur neuen Intervall-Untergrenze; ist sin(a) größer als x, so wird a zur neuen Obergrenze. Dann wird a als Mitte des Intervalls neu bestimmt. Versuche haben ergeben, daß die Genauigkeit im Bereich 0,8 < x < 9,999999E-1 bis zur dritten Nachkommastelle gegeben ist, bei kleineren Werten wird sie wesentlich besser. Dies liegt wohl an der in der LIBF.A-Bibliothek implementierten Sinus-Funktion. Versuche, den arcsin durch Reihenentwicklung zu bilden, ergaben gleich gute Genauigkeiten, jedoch bei x > 0,9 unzumutbar lange Rechenzeiten: Bei x=0,999999 etwa 30 Sekunden! Eine Romberg-Integration der Ableitung von arcsin(x) ist zwar sehr schnell, jedoch auch sehr ungenau.
Die im Bild gezeigte Datei muß den Namen fpasin.c erhalten und zu fpasin.o übersetzt werden. Diese Objekt-Datei wird dann mit dem Archiv-Utility in die Bibliothek LIBF.A eingebunden, und zwar nach der Datei ftoffp.o. Die dazu nötige Befehlsfolge lautet:

ar68 rav (oder ra) ftoffp.o libf.a fpasin.o

Anzumerken ist, daß die Funktionen sin(x) und cos(x) zwar mit sin() und cos() aufgerufen werden, in LIBF.A jedoch unter den Namen fpsin.o und fpcos.o zu finden sind. Der Aufruf der neuen Routine ist demnach auch asin(x) und nicht etwa fpasin(x). Die im Handbuch erwähnten Funktionen tan(x) und atan(x) gibt es übrigens weder in LIBF.A noch in CLIB.

Hinsichtlich der im Programm verwendeten Funktionen gibt das CP/M-Handbuch Auskunft. Für die der Routine übergenen Argumente gibt es keine Einschränkung, außer daß es Float-Argumente sein sollten. Der Funktionswert wird in jedem Fall so berechnet, daß er einem Argument im Bereich -1...+1 entspricht.

Reinhard Boenigk

```
float asin(ein)
float ein;
{ float a,floor(),fabs(),sin(),zw,u = 0.,vergl = 0., pi = 3.14159265359;
  int flag;  /*zw: sin(a) zum Vergleich mit vorherigem sin(a) = vergl.*/
  a = pi/4.;
  o = pi/2.;
  flag = 0;   /*flag für negatives Vorzeichen des Arguments, geloescht*/
  if (ein == 0. || ein == -0.)/*Ulkigerweise unterscheidet C +0 und -0*/
     return (0.);
  if (ein < 0.)                    /*Ausfuehrung bei negativem Argument*/
     { flag = 1;
       ein  = fabs(ein);                    /*Absolutwert des Arguments*/
     }
  ein-=floor(ein);          /*Argument auf den Bereich 0...1 begrenzen*/
  if (ein == 0.)  /*Falls das Argument ganzzahlig war, ist es nun Null*/
     return(flag ? -pi/2. : pi/2.);
  for (;;)                                          /*Endlosschleife*/
     { zw = sin(a);
       if (zw == vergl)                            /*Abbruch-Bedingung*/
          return (flag ? -a : a);
       if (zw > ein)
            { o = a;
              a = u + (a - u)/2.;
            }
       else { u = a;
              a += (o - a)/2.;
            }
       vergl = zw;            /*sin(a) zum Vergleich mit naechstem Wert*/
       continue;               /*Naechster Durchlauf der Endlosschleife*/
     }
}
```

Nach Einbinden dieses Programmes in LIBF.A hat man den arcsin immer zur Verfügung

Michael Schummel

Grenzgänger

Der AT und die 1-MByte-Grenze

Das BIOS-ROM in den ATs ist bereits darauf vorbereitet, die 1-MByte-Grenze der XTs und des MS-DOS zu überwinden. Diese Möglichkeit kann mit wenig Aufwand selbst in höheren Programmiersprachen genutzt werden.

In jedem IBM-AT und AT-Kompatiblen schlägt ein 80286-Herz. Dieser Prozessor hat 24 Adreßleitungen, mit denen er 16 MByte adressieren kann. Damit er zu dem 8088/8086-Prozessor der XTs kompatibel ist, hat der 80286 zwei verschiedene Betriebszustände erhalten. Nach einem Reset und während der MS-DOS-Aktivitäten arbeitet der Prozessor im „Real Address Mode", in dem er die Speicheradressen wie ein 8088/8086 berechnet. Dabei bleiben aber auch die vier oberen Adreßleitungen immer auf 0. Mit den verbleibenden 20 Adreßleitungen kann er dann eben nur 1 MByte ansprechen.
Eine Umschaltung des Prozessors in seinen „Virtual Protected Address Mode" hebt diese Beschränkung zwar auf, aber die Berechnung der Speicheradresse läuft in dieser Betriebsart ganz anders ab.
Ein AT läßt sich mit Speicherkarten bestücken, auf denen viele Megabytes RAM im Bereich oberhalb der 1-MByte-Grenze bestückt sein können. Dieser Speicherbereich, der von Adresse 100000H bis FFFFFFH reichen kann (15 MByte), wird als Extended Memory bezeichnet.
Dieses Extended Memory der ATs kann, im Gegensatz zu dem sogenannten „Expanded Memory" für XTs, nicht durch verschiedene Bankswitching-Verfahren (LIM-EMS, AQA-EMS) erreicht werden, sondern nur durch direkte Adressierung. Dazu muß sich der 80286 im „Protected Virtual Address Mode" befinden. (Allerdings können manche AT-Speichererweiterungskarten wahlweise mit dem Speicherbank-Umschaltungsverfahren oder mit direkter Adressierung zugreifen.)
Das hört sich im Moment etwas umständlich an, ist es aber gar nicht. Das ROM-BIOS der ATs enthält nämlich Routinen, die Programmierern unter MS-DOS bei der Nutzung des Extended Memory entgegenkommen.

Der Interrupt 15H

Im XT-ROM-BIOS ist der Interrupt 15H mit AH=0...3 für Kassetten-Ein-/-Ausgabe definiert. Diese Funktionen gibt es im AT-ROM-BIOS nicht mehr. Dafür sind im Interrupt 15H die Funktionen 80H bis 91H definiert worden. Einige dieser Funktionen werden im ROM-BIOS ohne weitere Operationen mit einem Rücksprung (IRET) abgeschlossen. Sie werden wohl erst im BIOS späterer Multitasking-Betriebssysteme ausgefüllt.

Die folgenden Funktionen sind aber schon heute in jedem AT sinnvoll nutzbar:

INT 15H:
AH=83H: Warten auf ein Ereignis
AH=84H: Joystick-Unterstützung
AH=86H: Warten in µs
AH=87H: Blocktransfer im 16-MByte-Adreßbereich
AH=88H: Größe des Extended Memory holen
AH=89H: In den Protected Virtual Mode umschalten

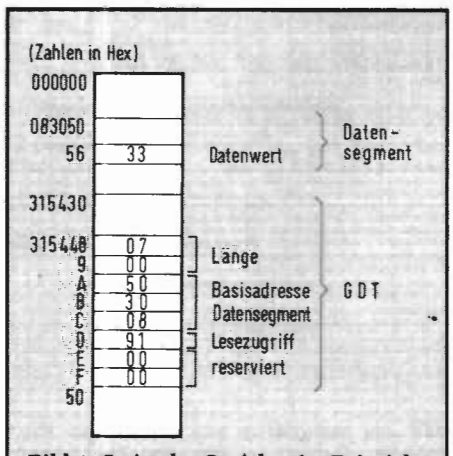


Bild 1. So ist der Speicher im Beispiel belegt

Dieser Beitrag befaßt sich im wesentlichen mit der Funktion 87H und kurz mit der Funktion 88H.
Es ist auch unter MS-DOS 3.x nicht möglich, mehr als 640 KByte RAM-Speicher direkt zu nutzen. Mit der Funktion 87H des INT 15H kann man besonders schnell momentan nicht benötigte Daten im Extended Memory blockweise vorübergehend auslagern. Durch diese ROM-BIOS-Funktion ist das etwa so leicht, wie eine einfache Diskettenausgabe, nur viel schneller, auch schneller als eine RAM-Disk-Operation.
Im übrigen basiert das Programm VDISK.SYS des MS-DOS 3.x unter Anwendung des Parameters /E auf eben dieser BIOS-Funktion, um eine RAM-Disk im Extended Memory zu organisieren; allerdings kommt zu dem eigentlichen Datentransfer noch die Organisation des Inhaltsverzeichnisses hinzu.

Adreßrechnung

Soweit es für das Verständnis des Interrupts 15H benötigt wird, sollen jetzt die Unterschiede zwischen den beiden Adreßmodi des 80286 bei der Adreßrechnung dargestellt werden.
Sicher geht das besonders gut an einem einfachen Beispiel. Zu diesem Beispiel ist im *Bild 1* eine Speicherbelegung dargestellt. Sie gilt für beide Modi.
Wir wollen in jedem Mode mit dem Assemblerbefehl
MOV AL, [0006]
das Byte mit dem Wert 33H an der absoluten Adresse 083056H in das AL-Register laden.

Real Address Mode

Vor Ausführung dieses Befehls muß das Segmentregister DS mit dem Wert 8305H geladen werden. Vom 8088/8086 her ist vielen schon bekannt, wie die Adreßbildung beim Aufruf des Befehls abläuft: Der Inhalt des DS-Registers wird mit 10H(16) multipliziert, was einer Verschiebung des DS-Registers um 4 Bit nach links entspricht. Dazu wird die Operandenadresse aus dem Befehl addiert. So entsteht die absolute (physikalische) Adresse:

8305H*10H = 83050H + 0006 = 83056H

Das Ergebnis ist immer 20 Bit lang, die oberen 4 Bit der 80286-Adresse bleiben

somit auf 0. Damit ist wieder einmal die 1-MByte-Grenze aufgezeigt.

Protected Virtual Address Mode

Hier ist der Ablauf komplizierter: Vor Ausführung des Beispiel-Befehls muß eine Global Descriptor Table (GDT) aufgebaut worden sein (im Beispiel ab Adresse 315430H). Das 40-Bit-CPU-Register GDTR muß mit der Startadresse und mit der Länge der GDT geladen werden. Im Beispiel sind diese fünf Bytes 3154300020H. Dabei ist 0020H die Länge der GDT in Bytes. Für die Basis-Adresse der GDT sind 24 Bit (3 Byte) Platz, d. h., die GDT kann irgendwo im 16-MByte-Adreßbereich liegen. Die Länge ist auf maximal 64 KByte begrenzt.
Die GDT kann maximal 8192 acht Byte lange Descriptoren (Bezeichner) unterbringen. Diese acht Byte langen Descriptoren sind zwar alle ähnlich aufgebaut, sie können aber mehrere unterschiedliche Bedeutungen haben. Uns interessiert hier nur der Typ „Segment-Descriptor" (*Bild 2*). Ein Segment-Descriptor wird zur Bildung physikalischer Adressen in Verbindung mit einem Segmentregister benutzt.
In dem Beispiel haben wir den vierten Descriptor der GDT mit den Werten für die Datenbereichsadresse geladen. Es hätte auch jeder andere der 8192 Descriptoren sein können.
Die acht Byte eines Segment-Descriptors haben folgende Bedeutung:

- Die niederwertigen zwei Byte erhalten die Länge des Datenbereichs in Byte.
- Die nächsten drei Byte enthalten die 24-Bit-Basis-Adresse des Datenbereichs.
- Das sechste Byte enthält Angaben über Zugriffsrechte zu dem Datenbereich.
- Die obersten zwei Byte sind für den 80386 reseviert und sollten für den 80286 immer auf 0 gesetzt werden.

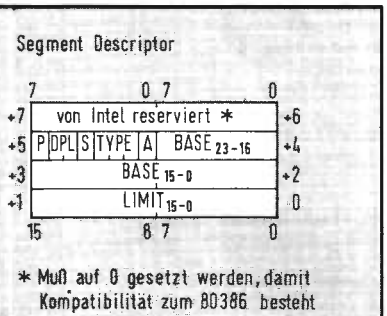


Bild 2. Ein Segment-Descriptor besteht aus acht Byte

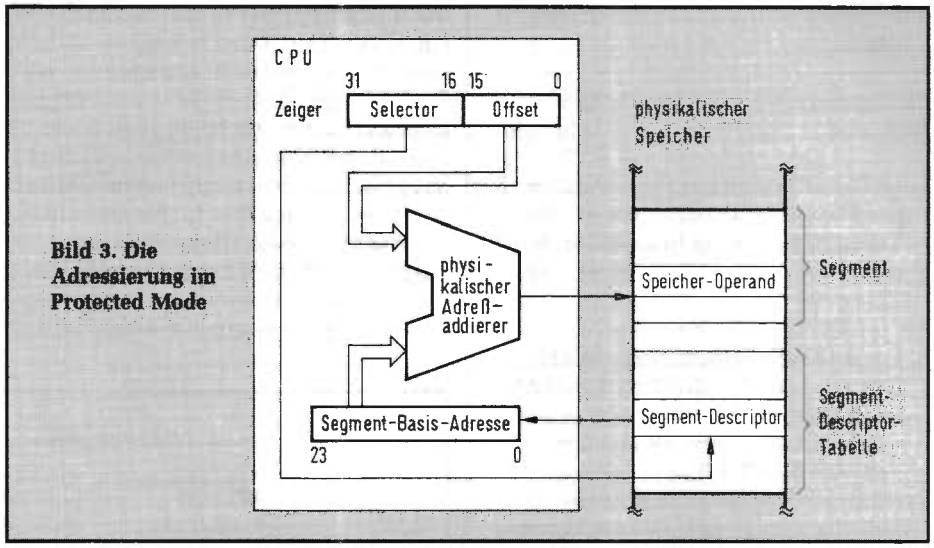


Bild 3. Die Adressierung im Protected Mode

Bei der Adreßbildung hat das DS-Register nun nicht mehr die Aufgabe, einen Teil der physikalischen Adresse direkt zu geben, sondern es ist jetzt ein Zeiger auf den Descriptor, der die Basisadresse des gewünschten Datenbereichs enthält. Die Adreßberechnung bei Ausführung des Befehls im Beispiel verläuft deshalb wie folgt (siehe auch *Bild 3*):
Der 16-Bit-Inhalt des DS-Registers (0018H) wird zu der 24-Bit-Basisadresse der GDT im GDTR (315430H) addiert. Das Ergebnis zeigt auf den Segment-Descriptor, der die Basisadresse und die Länge des Datenbereichs enthält. Zu dieser 24-Bit-Basisadresse (083050H) wird der 16-Bit-Operand unseres Befehls (0006H) addiert. Das Ergebnis ist die physikalische 24-Bit-Adresse (083056H) des Datenwertes.

DS + GDTR = Zeiger auf Descriptor
Descriptorinhalt + Operand = physikalische Adresse

Die unterschiedlichen Bedeutungen der Segment-Register und die damit zusammenhängende Adreßrechnung zeigt an dieser Stelle deutlich, warum derzeitige MS-DOS-Versionen auf keinen Fall im Protected Virtual Mode laufen können. Weitere, hier nicht dargestellte Unterschiede verschlimmern die Situation noch mehr.

Vorbereitungen

Von einem Programm aus, das unter MS-DOS läuft, wollen wir die Funktion 87H des Interrupts 15H als Tor zum Extended Memory nutzen. Deshalb müssen wir uns jetzt mit den Vorbereitungen zum Aufruf des Interrupts 15H beschäftigen. Wie oben angesprochen, kann mit der Funktion 87H ein zusammenhängender Datenblock einer beliebigen Adresse an eine andere beliebige Adresse kopiert werden. Die Quell- wie auch die Ziel-Adresse kann irgendwo im 16-MByte-Adreßbereich liegen. Das bedeutet, diese Funktion ermöglicht den Datentransfer vom MS-DOS-RAM in das Extended Memory und zurück.
Die Funktion 87H des Interrupts 15H nimmt dem Programmierer viel Arbeit bei der Vorbereitung des Transfers ab.

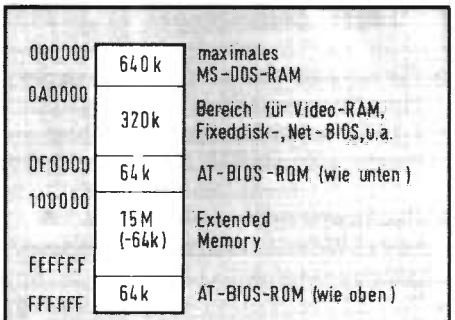


Bild 4. Speicherorganisation beim AT

Insbesondere mit dem Laden der speziellen 80286-Register GDTR und IDTR (Interrupt Descriptor Table Register) sowie dem Umschalten zwischen dem Real Address Mode und dem Protected Virtual Address Mode braucht sich der Programmierer nicht zu beschäftigen. Dem Anwender dieser BIOS-Funktion muß aber folgendes bekannt sein:

- Die Größe des installierten Extended Memory und dessen Nutzung (siehe weiter unten);
- Quell- und Zieladresse für den Daten-Transfer;

– Länge des zu transferierenden Datenblocks.

Im Detail sieht die Vorbereitung wie folgt aus:

1. Im Datenbereich des Programms muß der Platz für eine Global Descriptor Table (GDT) mit sechs Descriptoren reserviert werden. Das sind 48 (30H) Bytes.

2. Der dritte Descriptor muß mit den Informationen über den Quell-Datenbereich geladen werden (siehe oben):
   – 2 Byte Länge (am einfachsten 0000 = 64 kByte);
   – 3 Byte physikalische Adresse des niederwertigsten Bytes des Datenblocks;
   – 1 Byte Zugriffsrecht (nur lesen = 91H);
   – 2 Byte reserviert (0000 für 80386-Kompatibilität).

3. Der vierte Descriptor muß mit den Informationen über den Ziel-Datenbereich geladen werden:
   – 2 Byte Länge (0000);
   – 3 Byte physikalische Adresse des Bytes, an die das niederwertigste Byte des Datenblocks geladen werden soll;
   – 1 Byte Zugriffsrecht (schreiben/lesen = 93H);
   – 2 Byte (0000).

4. Die Basisadresse der GDT muß in der Form Segment:Offset in die Register ES:SI geladen werden (Assembler-Befehl: LES SI,GDT_Basis).

5. Die Anzahl der zu transferierenden 16-Bit-Worte muß in das CX-Register geladen werden.

6. Das Register AH wird mit der Funktionsnummer 87H geladen.

Jetzt kann der Interrupt 15H aufgerufen werden und der Blocktransfer wird durchgeführt.
Es ist für die Funktion unbedeutend, aber wer sich nach Rückkehr aus dem Interrupt 15H die GDT ansieht, kann feststellen, daß der zweite, fünfte und sechste Descriptor auch gefüllt sind. Diese Werte können für den nächsten Aufruf des Interrupts 15H unverändert stehenbleiben.

Randbedingungen

Da in den Descriptoren die Längenangabe für ein Segment 16 Bit groß ist, kann der Blocktransfer je Aufruf maximal 64 KByte bewegen. Das CX-Register enthält aber die Anzahl der zu kopierenden 16-Bit-Worte, deshalb darf dieser Wert maximal 32 KWorte (8000H) groß werden. Die transferierte Blocklänge hängt hauptsächlich von dem Wert im CX-Register ab. Die Angaben in den Descriptoren werden nur für Überprüfungen durch den 80286 benutzt. Sie müssen deshalb groß genug sein, sonst gibt es einen prozessorinternen Fehler (Trap).

Das ist eine besondere Eigenschaft des Protected Mode.
Bei dem Transfer langer Blöcke (mehr als etwa 5 KByte) muß noch das folgende Problem beachtet werden: Während des gesamten Blocktransfers befindet sich der 80286 im Virtual Protected Address Mode. Das BIOS sorgt dafür, daß in dieser Zeit alle Interrupts nicht wie gewohnt abgearbeitet, sondern nur mit einem einfachen IRET beendet werden. Es dürfen während des Transfers also keine

```
program extended_memory_beispiel;
const
   Blocklaenge = 512;
   readonly    = $91;
   readwrite   = $93;
type
   Reg = record
           AX,BX,CX,DX,BP,
           SI,DI,DS,ES,Flags : Integer;
         end;
   AbsAdr      = record
                    Low8Bit   : Byte;
                    High16Bit : Integer;
                 end;
   Descriptor  = record
                   SegmentLaenge : Integer ;
                   SegmentBasis  : AbsAdr ;
                   ZugriffRecht  : Byte ;
                   Frei          : Integer ;
                 end;
   GDT_Typ = array [1..6] of Descriptor;
   Datapointer = ^DataArray ;
   DataArray   = array [0..7,0..31] of integer ;
var
   XMStackpointer  : Integer;
   Beispielarray1,
   Beispielarray2  : DataArray;
{=============================================}
function XMStackNeu : Integer;
   var
      Register : Reg;
   begin
      Register.AX := $8800;
      Intr($15,Register);
      writeln(Register.AX,
              ' kBytes Extended Memory');
      writeln;
      XMStackNeu := (Register.AX + 1024)shl 1;
   end;
{=============================================}
procedure pushxm (Pointer : Datapointer);
   var
      GDT : GDT_Typ;
      Register : Reg;
   begin
      with GDT[3] do
      begin
         SegmentLaenge := Blocklaenge;
         SegmentBasis.Low8Bit :=
                  Seg(Pointer^) shl 4 +
                  Ofs(Pointer^);
         SegmentBasis.High16Bit :=
                  Seg(Pointer^) shr 4 +
                  Ofs(Pointer^) shr 8;
         ZugriffRecht := readonly;
         Frei := 0;
      end;
      XMStackPointer := pred(XMStackPointer);
      writeln('Array zwischengespeichert ab');
      writeln('Adresse ',
              XMStackPointer*512.0:7:0);
      writeln;
```

Bild 5. Mit Turbo-Pascal jenseits der 1-MByte-Grenze

zeitkritischen Interrupts auftreten. Werden zum Beispiel bei einem Disk-E/A die Interrupts nicht schnell genug bedient, gehen Daten verloren. Das ist bei der Festplatte besonders kritisch. Wer lange Blöcke bewegen möchte, sollte immer sicher sein, daß alle Disk-Operationen abgeschlossen sind, keine Kommunikation über Ports auflaufen kann und kein Disk-Druckerspooler tätig ist. Bei Blocklängen von nur wenigen 100 Byte werden allerdings keine Probleme auftreten.

Programmierer aufgepaßt!

Jetzt noch ein paar Hinweise zur Aufteilung des Extended Memory (siehe auch *Bild 4*). Ein MS-DOS-Programm kann die Größe des Extended Memory über den Interrupt 15H mit AH=88H abfragen. Bei Rückkehr steht die Größe in KByte im AX-Register. Mit einem Extended Memory von 512 KByte wird AX=0200H.
Das Extended Memory kann durch eine RAM-Disk (z. B. VDISK.SYS) genutzt werden. Solche RAM-Disks liegen am unteren Ende des Extended Memory wenn nicht der gesamte Speicher genutzt wird. Die Aufteilung des Speichers liegt voll in der Verantwortung des Programmierers, da eine RAM-Disk-Software nicht merkt, wenn ein Programm in den durch sie genutzten Speicherbereich schreibt.
Wenn zum Beispiel in einem Extended Memory von 512 KByte Größe eine RAM-Disk von 400 KByte Größe eingerichtet wird, belegt die RAM-Disk den Speicher von Adresse 100000H bis 163FFFH. Der Speicher zwischen 164000H und 17FFFFH steht für den Programmierer zur Verfügung. Zur Erinnerung: die 640 KByte für MS-DOS liegen zwischen 000000H und 09FFFFH.

```
            with GDT[4] do
            begin
               SegmentLaenge := Blocklaenge;
               SegmentBasis.Low8Bit := 0;
               SegmentBasis.High16Bit :=
                              XMStackPointer shl 1;
               ZugriffRecht := readwrite;
               Frei := 0;
            end;
            with Register do
            begin
               ES := Seg(GDT);
               SI := Ofs(GDT);
               CX := Blocklaenge shr 1;
               AX := $8700;
            end;
            Intr($15,Register);
         end;
      {================================================}
      procedure popxm (Pointer : Datapointer);
         var
            GDT : GDT_Typ;
            Register : Reg;
         begin
            with GDT[3] do
            begin
               SegmentLaenge := Blocklaenge;
               SegmentBasis.Low8Bit := 0;
               SegmentBasis.High16Bit :=
                           XMStackPointer shl 1;
               ZugriffRecht := readonly;
               Frei := 0;
            end;
            XMStackPointer := succ(XMStackPointer);
            with GDT[4] do
            begin
               SegmentLaenge := Blocklaenge;
               SegmentBasis.Low8Bit :=
                        Seg(Pointer^) shl 4 +
                        Ofs(Pointer^);
               SegmentBasis.High16Bit :=
                        Seg(Pointer^) shr 4 +
                        Ofs(Pointer^) shr 8;
               ZugriffRecht := readwrite;
               Frei := 0;
            end;
            with Register do
            begin
               ES := Seg(GDT);
               SI := Ofs(GDT);
               CX := Blocklaenge shr 1;
               AX := $8700;
            end;
            Intr($15,Register);
         end;
      {================================================}
      begin
         XMStackPointer := XMStackNeu;
         Beispielarray1[3,5] := 12345;
         PushXM(Addr(Beispielarray1));
         PopXM(Addr(Beispielarray2));
         writeln(Beispielarray2[3,5],
                 ' = transferierter Wert');
      end.
```

Beispielprogramm

Bild 5 zeigt ein Anwendungsbeispiel für die bis hier beschriebenen Funktionen. Das Turbo-Pascal-Programm richtet am oberen Ende des vorhandenen Extended Memory einen Stackbereich ein. Diese Initialisierung führt die Funktion XMStackNeu durch.
Die Prozedur PushXM transferiert einen 512 Bytes langen Datenblock in das Extended Memory. Die Prozedur PopXM transferiert einen 512 Bytes langen Datenblock aus dem Extended Memory zurück in das MS-DOS-Memory. Dabei wird jeweils gleichzeitig der XMStackpointer aktualisiert. Innerhalb der Prozeduren werden praktisch nur die oben dargestellten Vorbereitungen durchgeführt. Deshalb ist das Programm im wesentlichen selbsterklärend.
In diesem Beispiel wird das Extended Memory als Stackbereich, also als First-In/Last-Out-Puffer genutzt. Genauso wäre natürlich ein First-In/First-Out-Puffer denkbar, zum Beispiel für einen Drukkerspooler. Auch eine Datenübergabe zwischen verschiedenen, nacheinander laufenden Programmen ist denkbar. Wer sich besonders gut auskennt, kann auch mit Blöcken unterschiedlicher Länge und wahlfreiem Zugriff arbeiten. Er sollte dabei aber nicht den Überblick verlieren, denn das MS-DOS gibt bei all diesen Operationen keinerlei Hilfe, da es davon überhaupt nichts mitbekommt.

Literatur

[1] iAPX 286/10 Advance Information, Intel Corporation, 1983
[2] Technical Reference, Personal Computer AT, IBM, 1984

Karl-Friedrich Penning

Einfache Schwarzweiß-Grafik unter C

Die Programmiersprache C eignet sich sehr gut für Berechnungen im technisch-wissenschaftlichen Bereich. Leider fehlt aber bei den meisten Compilern die Möglichkeit, mathematische Ergebnisse grafisch darzustellen, was in der Technik unerläßlich ist. Auf dem Markt werden zwar „Graphic Toolboxes" angeboten, sie sind jedoch teuer und bieten meist mehr, als man in der Regel zur Darstellung einfacher Kurvenzüge benötigt.

```
/*            Einfache s/w-Grafik unter 'C'              */
/*********************************************************/
/*
**    Vers. 2.1       11.2.87
**    Autor: Karl-Fr. Penning
*/

#include <dos.h>
#include <stdlib.h>

#define SCHIRM  16          /*  Bildschirminterrupt       */
#define HRG      6          /*  High Resolution Grafics   */
#define TEXT     2          /*  Text-Bildschirm           */


static int xalt,yalt;       /*  Merker fuer letzten Punkt */


void grafset()              /*  Bildschirm auf hochauflösende
                            Grafik 640 * 200 umstellen    */
{
    union REGS cpureg;

    cpureg.h.ah  = 0;       /*  Video-Modus               */
    cpureg.h.al  = HRG;     /*  Grafik                    */
    int86( SCHIRM , &cpureg , &cpureg );
}


void textset()              /*  Bidschirm auf Textdarstellung
                            umschalten                    */
{
    union REGS cpureg;

    cpureg.h.ah  = 0;       /*  Video-Modus               */
    cpureg.h.al  = TEXT;    /*  Textdarstellung           */
    int86( SCHIRM , &cpureg , &cpureg );
}


void punkt(x,y)             /*  Punkt in Position x,y setzen */
    unsigned x,y;           /*  0,0 ist links oben           */
{
    union REGS cpureg;

    cpureg.h.ah  = 12;      /*  Punkt schreiben           */
    cpureg.h.al  = 1;       /*  Farbcode                  */
    cpureg.x.dx  = (yalt = y);
    cpureg.x.cx  = (xalt = x);
    if(x<0 || y<0 || x>639 || y>199)
        return;
    int86( SCHIRM , &cpureg , &cpureg );
}


void line(x1,y1,x2,y2)      /*  Linie zeichnen von x1,y1 */
    int x1,y1,x2,y2;        /*  nach x2,y2               */
{
    long dx,dy;
    int i;

    dx = x2 - x1;           /* laengere Achse ermitteln  */
    dy = y2 - y1;
    if(abs(dx) > abs(dy))   /* und relativ dazu zeichnen */
        if(dx>0) for( i=0 ; i<=dx ; i++ )
            punkt(x1+i,(int) (y1+((dy*i*2)/dx+1)/2));
            else for( i=0 ; i>=dx ; i-- )
                punkt(x1+i,(int) (y1+((dy*i*2)/dx+1)/2));
        else
            if(dy>0) for( i=0 ; i<=dy ; i++ )
                punkt( (int) (x1+((dx*i*2)/dy+1)/2),y1+i);
            else for( i=0 ; i>=dy ; i-- )
                punkt( (int) (x1+((dx*i*2)/dy+1)/2),y1+i);
    }


    void lineto(x,y)            /* Linie vom letzten         */
        unsigned x,y;           /* gezeichneten Punkt nach   */
    {                           /* x,y zeichnen              */
        line(xalt,yalt,x,y);
    }
```

Bild 1. Quelltext der Grafik-Prozeduren

```
/*            Einfache s/w-Grafik unter 'C'              */
/*********************************************************/
/*
**    Vers. 2.1       11.2.87
**    Autor: Karl-Fr. Penning
**
**       INCLUDE - File
**  Deklaration der Grafik-Routinen aus GRAFIK.LIB
*/

void grafset(void);         /*  Bildschirm auf hochauflösende
                            Grafik 640 * 200 umstellen    */

void textset(void);         /*  Bidschirm auf Textdarstellung
                            umschalten                    */

void punkt(int,int);        /*  Punkt in Position x,y setzen */
                            /*  0,0 ist links oben           */

void line(int,int,int,int); /*  Linie zeichnen von x1,y1  */
                            /*  nach x2,y2                */

void lineto(int,int);       /*  Linie vom letzten         */
                            /*  gezeichneten Punkt nach   */
                            /*  x,y zeichnen              */

/*    E n d e    */
```

Bild 2. Mit dieser Include-Datei kann man die Prozeduren deklarieren

Die hier vorgestellte kleine Grafik-Bibliothek wurde erstellt, um z. B. Meßwertkurven und Diagramme in hochauflösender Grafik auf IBM-kompatiblen PCs darstellen zu können. Das wird möglich durch fünf Funktionen:

grafset( )

Diese Funktion schaltet auf Grafikdarstellung um und löscht den Bildschirm.

textset( )

Diese Funktion schaltet auf Textdarstellung (80 Zeichen) zurück und löscht den Bildschirm.

punkt(x,y)

Über die beiden Integer-Variablen x und y wird dieser Funktion die Position eines zu setzenden Punktes mitgeteilt. Bei der Standard-IBM-Grafik werden x-Werte im Bereich von 0 bis 639 und y-Werte im Bereich von 0 bis 199 akzeptiert. x und y dürfen auch außerhalb dieses Bereiches liegen; ein Punkt wird dann jedoch nicht gezeichnet. Der Punkt mit den Koordinaten 0,0 ist oben links.

line(x1,y1,x2,y2)

Diese Funktion zeichnet eine Gerade zwischen den Punkten x1,y1 und x2,y2. Die Punkte dürfen auch außerhalb des Darstellbereiches liegen. Es wird dann nur der im Darstellungsbereich liegende Geradenteil gezeichnet.

lineto(x,y)

Von dem zuletzt angesprochenen Punkt, der auch außerhalb des Darstellbereiches liegen kann, wird eine Gerade zum Punkt mit den Koordinaten x,y gezeichnet.

Den Quelltext zeigt *Bild 1*. Will man die Funktionen anwenden, muß man sie zunächst deklarieren. Das kann über ein Include-File geschehen, wie in *Bild 2* gezeigt. *Bild 3* zeigt ein Anwendungsbeispiel und *Bild 4* das Resultat.

```
/*
**          G R A F T S T
**
**  Vers.: 1.0      11.2.87
**  Autor: Karl-Fr. Penning
**
**  Test für die Grafik-Bibliothek GRAFIK.LIB
*/

#include <stdio.h>
#include <conio.h>
#include <grafik.h>
#include <math.h>

#define PI2  6.2831852      /*  2 Pi  */

/* Vorwärtsdeklaration */
float funktion(int);

main()
{
        int i,x,y;
        float wert;

        printf("\n\n\
                        G R A F T E S T\n\n\
Dieses Grafik-Testprogramm zeichnet zunächst einen Oszilloskop-ähnlichen\n\
Rahmen und trägt eine Sinuskurve ein.\n\n\
    - Taste drücken - ");
        getch();

        grafset();              /* grafische Darstellung einschalten */

        /*  Oszilloskop-ähnliche Darstellung zeichnen  */

        for( y=0 ; y<181 ; y+=18 )        /* horizontale punktierte */
            for( x=135 ; x<636 ; x+=3 )   /* Linien zeichnen        */
                punkt(x,y);

        for( x=135 ; x<636 ; x+=50 )      /* vertikale punktierte   */
            for( y=0 ; y<181 ; y+=3 )     /* Linien zeichnen        */
                punkt(x,y);

        line(135,0,135,180);              /* durchgezogene Umrandung */
        lineto(635,180);                  /* zeichnen                */
        lineto(635,0);
        lineto(135,0);

        line(125,90,639,90);              /* Null-Linie              */

        /*  Funktionswerte eintragen  */

        wert = funktion(0);
        punkt( 135 , 90 - 80.0 * wert );
        for( i=1 ; i<500 ; i++ )
        {
            wert = funktion(i);
            lineto( 135+i , 90 - 80.0 * wert );
        }

        /* auf Tastendruck warten  */
        printf("\n\n\n\n\n\n TASTE\ndruecken\n");
        getch();
        textset();
}

/*******************************************************/

float funktion(phi)             /* Testfunktion */
        int phi;
{
        return( sin( PI2 * phi / 500.0 ) );
}

/*************** E N D E  ******************************/
```

Bild 3. Dieses Beispiel erzeugt die Grafik von Bild 4

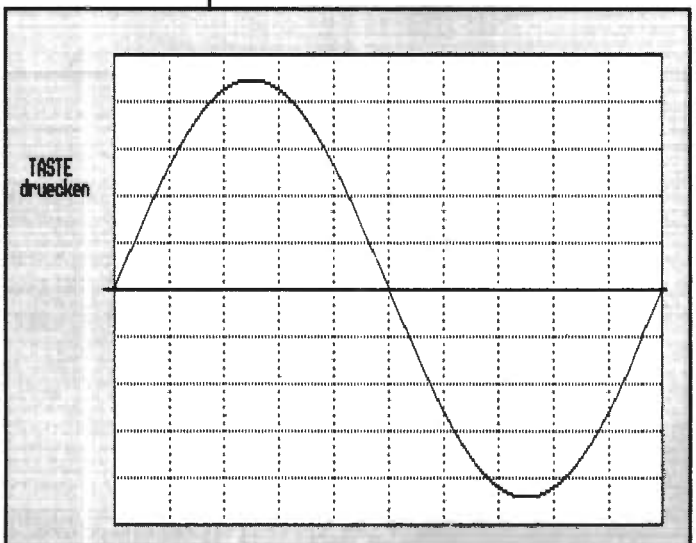


Bild 4. Das Ergebnis des Beispielprogramms von Bild 3

Holger Haase, Michael Menrad

mc-AllProg

PALs, EPROMs, PROMs ... auf einem Gerät programmieren, Teil 3

In diesem letzten Teil der Serie über ein universell einsetzbares Programmiergerät geht es um die Beschreibung der Host-Software. Also um den Teil, der letztendlich dafür verantwortlich ist, wie effektiv und komfortabel man mit dem ganzen arbeiten kann. Das zu AllProg gehörende Programmpaket ist in Versionen für CP/M (ab Version 2.2) und PC-DOS verfügbar.

Das Paket ist vollständig in Turbo-Pascal geschrieben und stellt folgende Anforderungen an Ihren Rechner:

Unter CP/M:

Z80-CPU mit beliebiger Taktrate.
Freier Arbeitsspeicher von 100H... C000H.
Bildschirmdarstellung mit mindestens 24 × 80 Zeichen.
Eine serielle Schnittstelle, die über den READER- und PUNCHER-Kanal anzusprechen ist.

Unter PC-DOS:

128 KByte freier Arbeitsspeicher.
Bildschirmdarstellung mit mindestens 24 × 80 Zeichen.
Eine serielle Schnittstelle an den Port-Adressen 03F8H...03FFH (COM1) oder 02F8H...02FFH (COM2).

Ein Installationsprogramm dient zur Anpassung des Programms an das bei Ihnen vorhandene System. Sie können dabei aus einer Liste eins von 28 Terminals auswählen. Sollte das von Ihnen verwendete jedoch nicht dabei sein, so besteht die Möglichkeit, das Terminal durch Eingabe der benötigten Steuersequenzen zu installieren.

ALLPROG

Dies ist das Hauptprogramm, von dem aus alle weiteren Module des Programmsystems angesprochen werden. Nach dem Aufrufen durch „AP <RET>" meldet sich das Programm, wie in *Bild 1* dargestellt, mit dem Eröffnungsmenü.

Von hier aus wird, abhängig von der Bausteinart, die bearbeitet werden soll, in die nachstehend beschriebenen Module verzweigt. Da das Programmsystem von seinem Umfang her nicht vollständig in den Arbeitsspeicher paßt, wurde es in eine COM- und mehrere CHAIN-Dateien zerlegt. Wie sich die COM- und CHAIN-Dateien gegenseitig aufrufen zeigt *Bild 2*.

EPROM-PROG

Dieser Programmteil ermöglicht die Bearbeitung von EPROMs und EPROM-Dateien. Das EPROM-PROG-Menü wird, ebenso wie in PROM- und PAL-PROG, in drei Zeilen im oberen Bereich des Bildschirms eingeblendet (*Bild 3*). Da es mittlerweile EPROMs gibt, die eine Kapazität von 1 MBit besitzen, kann nicht der gesamte EPROM-Inhalt auf einmal im Arbeitsspeicher gehalten werden. Deshalb wird die Datei in Segmente zerlegt, die dann je nach Bedarf automatisch geladen bzw. gespeichert werden.

Die Kommandos von EPROM-PROG

Die Befehle zur Cursorsteuerung sind WordStar-kompatibel gehalten.

1. EPROM-Typ
Hier ist die Bezeichnung des zu behandelnden EPROMs einzugeben. Wird die Bezeichnung unvollständig eingegeben (fehlt z. B. die bei manchen EPROMs nötige Spannungsangabe), wird ein ähnlicher Typ zur Anzeige gebracht. Dieser kann nun mit <RET> bestätigt werden. Sollte der angezeigte Typ nicht dem gewünschten entsprechen, so kann man sich mit ↑ X weitere Vorschläge machen lassen.

2. Arbeitsdatei
Mit Hilfe dieser Funktion wird die Datei ausgewählt, die bearbeitet werden soll.

```
          mc - A l l P r o g

                  von

      Holger Haase & Michael Menrad,
          Peripheriegeräte GbR
             Postfach 2647
            3300 Braunschweig
            Tel.: 0531/346228

    ---------- Auswahl ----------

            1. EPROM - PROG
            2. PROM  - PROG
            3. PAL   - PROG

            9. Fehlerbericht
            0. ENDE

           >_<
```

Bild 1. Das Eröffnungsmenü

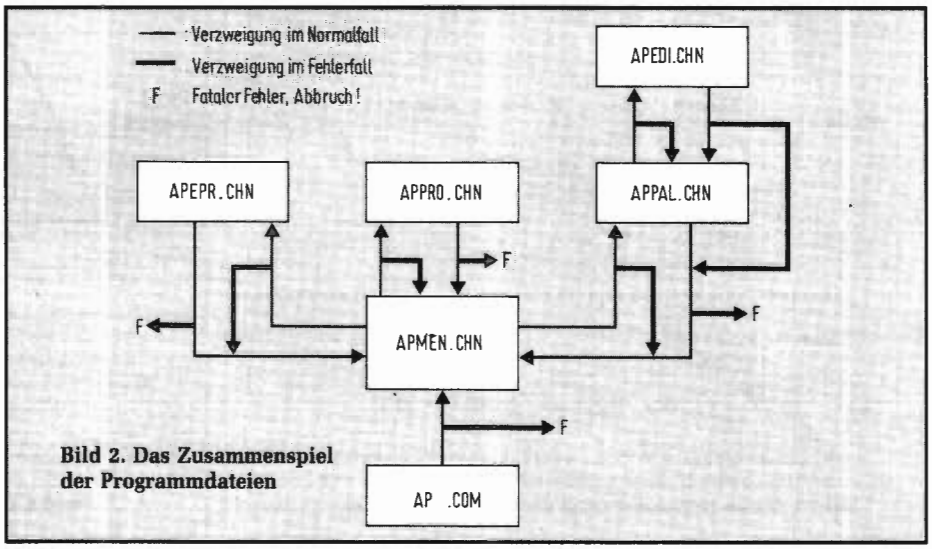


Bild 2. Das Zusammenspiel der Programmdateien

```
                    EPROM-PROG  V 2.0 (CP/M)          (C) Haase & Menrad
   max. Adr.  : 0FFFF                 letzte Adr.  : 00108
 1. EPROM-Typ : 27512 12.5V        2. Arbeitsdatei : BEISPIEL.EPR
 3. Editor       4. Lesen          5. Leertest       6. Programmieren
 7. Verifizieren                                     0. ENDE
>_
Adr. :  0  1  2  3   4  5  6  7    8  9  A  B   C  D  E  F    0123456789ABCDEF
-----+-----------------------------------------------------------------------
00010: 70 72 6F 6A  65 63 74 20   50 72 6F 6A  65 6B 74 2C    project Projekt,
00020: 20 50 6C 61  6E 2C 20 45   6E 74 77 75  72 66 3B 20     Plan, Entwurf;
00030: 56 6F 72 68  61 62 65 6E   2C 20 55 6E  74 65 72 6E    Vorhaben, Untern
00040: 65 68 6D 65  6E 3B 20 77   65 72 66 65  6E 2C 20 73    ehmen; werfen, s
00050: 63 68 6C 65  75 64 65 72   6E 2C 20 66  65 75 65 72    chleudern, feuer
00060: 6E 3B 20 76  6F 72 73 74   72 65 63 6B  65 6E 2C 20    n; vorstrecken,
00070: 76 6F 72 73  70 72 69 6E   67 65 6E 20  6C 61 73 73    vorspringen lass
00080: 65 6E 3B 20  70 72 6F 6A   69 7A 69 65  72 65 6E 3B    en; projizieren;
00090: 20 65 6E 74  77 65 72 66   65 6E 2C 20  70 6C 61 6E     entwerfen, plan
000A0: 65 6E 2C 20  76 6F 72 68   61 62 65 6E  3B 20 76 6F    en, vorhaben; vo
000B0: 72 73 70 72  69 6E 67 65   6E 2E FF FF  FF FF FF FF    rspringen.......
000C0: 41 6E 6F 74  68 65 72 20   68 6F 70 65  20 66 65 65    Another hope fee
000D0: 64 73 20 61  6E 6F 74 68   65 72 20 64  72 65 61 6D    ds another dream
000E0: 2C 20 61 6E  6F 74 68 65   72 20 74 72  75 74 68 20    , another truth
000F0: 63 61 6E 20  73 74 61 72   74 20 62 79  20 74 68 65    can start by the
00100: 20 6D 61 63  68 69 6E 65   2E FF FF FF  FF FF FF FF     machine........
```

Bild 3. EPROM-PROG bei der Arbeit

Dabei ist zu beachten, daß keine Backup-Datei angelegt wird, sondern sämtliche Änderungen, die mit dem Editor vorgenommen werden, direkt in die Arbeitsdatei geschrieben werden.

3. Editor
Um die Bearbeitung der Bausteine möglichst komfortabel zu gestalten, wurde ein „Hex-Editor" geschrieben. Dieser Editor arbeitet bildschirmorientiert und verhält sich in etwa so, wie man es von Text-Editoren wie WordStar her gewohnt ist. Das heißt, man kann sich innerhalb eines Bildschirmausschnitts frei in der zu bearbeitenden Datei bewegen, um die gewünschten Änderungen vorzunehmen. Änderungen können in Hex oder in ASCII erarbeitet werden. Im Anschluß an diese Hauptfunktionen werden die weiteren Befehle des Editors beschrieben.

4. Lesen
Diese Funktion liest den EPROM-Inhalt von EPROM-Adresse XXXXXH bis EPROM-Adresse YYYYYH, bei Adresse ZZZZZH beginnend, in die voreingestellte Datei.

5. Leertest
Überprüft, ob alle Speicherstellen des EPROMs „leer" sind.

6. Programmieren
Hiermit wird die Arbeitsdatei von Datei-Adresse XXXXXH bis Datei-Adresse YYYYYH ab Adresse ZZZZZH ins EPROM programmiert.

7. Verifizieren
Der Dateibereich von XXXXXH bis YYYYYH wird mit dem EPROM-Inhalt ab Adresse ZZZZZH verglichen.

0. ENDE
Rückkehr zum Hauptmenü.

Der Hex-Editor

Nun zu den Funktionen, die der Hex-Editor zur Verfügung stellt:

<TAB> : Zwischen Hex- und ASCII-Eingabe umschalten.
↑D : Cursor zum nächsten Byte.
↑S : Cursor zum vorherigen Byte.
↑X : Cursor in die nächste Zeile.
↑E : Cursor in die vorherige Zeile.
↑C : Cursor eine Seite vorwärts.
↑R : Cursor eine Seite zurück.
↑QR : Cursor zum ersten Byte der Datei.
↑QC : Cursor zum letzten Byte der Datei.
↑QJ : Cursor zu einem beliebigen Byte der Datei.
↑KC : Bereich kopieren.
↑KF : Bereich mit Konstante füllen.
↑KR : Eine Datei in die Arbeitsdatei einlesen.
↑KW : Einen Teil der Arbeitsdatei in eine Datei schreiben.
↑KD : Zurück ins EPROM-PROG-Menü.

PROM-PROG

Der Aufbau dieses Programmes entspricht im wesentlichen dem von EPROM-PROG, so daß hier nur kurz auf die Unterschiede eingegangen werden soll. Die maximale Größe der Bausteine, die bearbeitet werden können, ist auf 16 KByte beschränkt. Doch müssen hier auch Bausteine bearbeitbar sein, die weniger als 256 Byte (eine Bildschirmseite) fassen. Weiterhin mußten die Ein-/Ausgaberoutinen modifiziert werden, da es viele PROMs gibt, die weniger als 8 Bit pro Speicherstelle besitzen. Die von PROM-PROG unterstützten Bausteine sind schwerpunktmäßig die stark verbreiteten Typen von Signetics, Valvo oder auch Philips.

PAL-PROG

Unter diesem Programmpunkt findet man alles, was nötig ist, um PALs komfortabel bearbeiten zu können. Während der Arbeit mit PAL-PROG präsentiert sich der Bildschirm wie in Bild 4. Zuerst einmal zu den drei mächtigsten PAL-PROG-Teilprogrammen:

```
                          PAL-PROG  V 1.0 (CP/M)          (C) Haase & Menrad

 1. PAL-Typ    : 16X4              2. Arbeitsdatei : BEISPIEL.PAL
 3. Editor       4. Lesen          5. Leertest       6. Programmieren
 7. Verifizieren 8. Simulieren     9. Hilfsprogramme 0. ENDE
>_
-------------------------------------------------------------------------------
Arbeitsdatei : BEISPIEL.PAL         Zeile: 1     Spalte: 1            Einfügen
-------------------------------------------------------------------------------
PAL 16X4

PINS
CLK,   IO,  I1, BO, B1, B2, B3,  I2,  CLR, GND,
/OC, /LIO, /GE, A3, A2, A1, AO, /LE, /RIO, VCC

GLEICHUNGEN
IF (/I2* I1*/IO)  RIO = (AO)

/AO :=    /I2*/I1*/IO*(/AO)*/CLR
        +  /I2*/I1* IO*(/BO)*/CLR
        +  /I2* I1*/IO*(/A1)*/CLR
        +   I2* I1*    (/AO)*/CLR
       :+:  I2*/I1*/IO*/RIO
        +   I2* I1*/IO* RIO
```

Bild 4. Das Menü von PAL-PROG

PAL-Assembler

Der PAL-Assembler übersetzt eine Quelldatei, die die gewünschten Funktionen in Form Boolescher Ausdrücke enthält, in die zur PAL-Programmierung nötige Liste der zu programmierenden Sicherungen. Die Syntax der Datei weicht nicht sehr stark von der ab, die andere PAL-Assembler verarbeiten, so daß vorhandene Quellen leicht angepaßt werden können. Die Quelldatei muß folgendermaßen aufgebaut sein (siehe auch *Bild 5*):

Als erstes wird der PAL-Typ festgelegt. Dies geschieht durch das reservierte Wort „PAL", hinter dem dann der Typ des PAL-Bausteins, zum Beispiel „16R4", steht. Dahinter folgt, eingeleitet durch das Wort „PINS", eine Anschlußliste in der jedem Anschluß des PALs ein Name und eine Polarität zugeordnet wird. Ein Anschluß kann als low-aktiv oder als high-aktiv vereinbart werden. Low-aktiven Anschlüssen wird ein „/" vorangestellt. Die Namen der Anschlüsse müssen durch Kommata getrennt werden. Es ist darauf zu achten, daß die Spannungsanschlüsse mit „VCC" bzw. mit „GND" bezeichnet werden müssen. Soll ein Anschluß nicht benutzt werden, so ist dieser mit „NC" (not connec)(ted) zu bezeichnen. Nach der Anschlußliste kommen die Gleichungen für die PAL-Ausgänge. Bei programmierbaren Ein-/Ausgängen kann der eigentlichen Ausgangsgleichung ein durch „IF" eingeleiteter Term vorangestellt werden, durch den die Funktion des Anschlusses gesteuert wird. Fehlt dieser Term bei einem programmierbaren Ein-/Ausgang, so geht der PAL-Assembler davon aus, daß der Anschluß nur als Ausgang dienen soll (dies entspricht der Angabe „IF (VCC)").

Bei der Ausgangsgleichung gilt es zu beachten, daß bei PAL-Bausteinen mit low-aktiven Ausgängen die in der Gleichung angegebene Polarität entgegengesetzt zu der in der Anschlußliste vorgegebenen sein muß.

In der Gleichung können folgende Operatoren verwendet werden (in der Reihenfolge ihrer Bearbeitung):

„/" : *NICHT-Operator*
Dieser Operator dient zum Invertieren eines Signals. Er muß vor dem Signal stehen, das invertiert werden soll.

„*" : *UND-Operator*
Der UND-Operator wird benutzt, um zwei oder mehr Signale zu undieren. Der dabei entstehende Ausdruck wird Produktterm genannt.

„+" : *ODER-Operator*
Mit diesem Operator werden zwei oder mehr Produktterme und/oder Signale oderiert.

":+:" : *EXKLUSIV-ODER-Operator*
Dieser Operator exklusiv-oderiert zwei oder mehr Produktterme, bzw. mit dem ODER-Operator verknüpfte Produktterme und/oder Signale. Er steht nur bei einigen PAL-Typen zur Verfügung.

Im Anschluß an den Gleichungsblock kann nun noch ein Simulationsblock folgen. Sein Aufbau und seine Funktion werden unter „PAL-Simulator" noch näher beschrieben. Kommentare können im Quelltext an beliebigen Stellen erscheinen, solange sie nicht innerhalb eines Wortes plaziert werden. Ein Kommentar beginnt mit „/*" und endet mit „*/".

PAL-Disassembler

Der PAL-Disassembler besitzt die Aufgabe, einen vorgegebenen PAL-Inhalt in eine leichter lesbare Textdatei umzuformen. Die durch ihn erzeugte Datei weist genau das Format auf, das der PAL-Assembler verlangt, so daß sie mit seiner Hilfe wieder in die Sicherungsliste umgewandelt werden kann. Der Disassembler wird dazu benutzt, den Inhalt unbekannter PALs zu untersuchen. Dabei ist zu beachten, daß dies natürlich nur

```
PAL 16X4

/* 4-Bit Auf/Abwärts-Zähler mit Schieberegister und Vergleicher */

PINS
CLK,   IO,  I1, BO, B1, B2, B3,  I2,  CLR, GND,
/OC, /LIO, /GE, A3, A2, A1, A0, /LE, /RIO, VCC

GLEICHUNGEN
IF (/I2* I1*/IO)                                  /* rechts rausschieben          */
 RIO = (AO)

/AO :=    /I2*/I1*/IO*(/AO)*/CLR                  /* AO halten                    */
       +  /I2*/I1* IO*(/BO)*/CLR                  /* BO laden                     */
       +  /I2* I1*/IO*(/A1)*/CLR                  /* nach rechts schieben         */
       +   I2* I1*    (/AO)*/CLR                 /* halten                       */
      :+:  I2*/I1*/IO*/RIO                        /* nach links schieben          */
       +   I2* I1*/IO* RIO                        /* inkrementieren bei Überlauf  */
       +   I2* I1* IO* RIO                        /* dekrementieren bei Unterlauf */
       +   CLR                                   /* AO löschen                   */

/A1 :=    /I2*/I1*/IO*(/A1)*/CLR                  /* A1 halten                    */
       +  /I2*/I1* IO*(/B1)*/CLR                  /* B1 laden                     */
       +  /I2* I1*/IO*(/A2)*/CLR                  /* nach rechts schieben         */
       +   I2* I1*    (/A1)*/CLR                 /* halten                       */
      :+:  I2*/I1*/IO*(/AO)                       /* nach links schieben          */
       +   I2* I1*/IO*( AO)* RIO                  /* inkrementieren bei Überlauf  */
       +   I2* I1* IO*(/AO)* RIO                  /* dekrementieren bei Unterlauf */
       +   CLR                                   /* A1 löschen                   */

/A2 :=    /I2*/I1*/IO*(/A2)*/CLR                  /* A2 halten                    */
       +  /I2*/I1* IO*(/B2)*/CLR                  /* B2 laden                     */
       +  /I2* I1*/IO*(/A3)*/CLR                  /* nach rechts schieben         */
       +   I2* I1*    (/A2)*/CLR                 /* halten                       */
      :+:  I2*/I1*/IO*(/A1)                       /* nach links schieben          */
       +   I2* I1*/IO*( A1)*( AO)* RIO            /* inkrementieren bei Überlauf  */
       +   I2* I1* IO*(/A1)*(/AO)* RIO            /* dekrementieren bei Unterlauf */
       +   CLR                                   /* A2 löschen                   */

/A3 :=    /I2*/I1*/IO*(/A3)*/CLR                  /* A3 halten                    */
       +  /I2*/I1* IO*(/B3)*/CLR                  /* B3 laden                     */
       +  /I2* I1*/IO* /LIO*/CLR                  /* nach rechts schieben         */
       +   I2* I1*    (/A3)*/CLR                 /* halten                       */
      :+:  I2*/I1*/IO*(/A2)                       /* nach links schieben          */
       +   I2* I1*/IO*( A2)*( A1)*( AO)* RIO     /* inkrementieren bei Überlauf  */
       +   I2* I1* IO*(/A2)*(/A1)*(/AO)* RIO     /* dekrementieren bei Unterlauf */
       +   CLR                                   /* A2 löschen                   */

IF ( I2)
 LIO =    I2*/I1*/IO*( A3)                       /* links rausschieben           */
       +  I2* I1*/IO*( A3)*( A2)*( A1)*( AO)* RIO /* Überlauf anzeigen           */
       +  I2* I1* IO*(/A3)*(/A2)*(/A1)*(/AO)* RIO /* Unterlauf anzeigen          */

 LE =     (A3*/B3)                               /* B3 kleiner A3                */
       +  (A3:*:B3)*(A2*/B2)                     /* B2 kleiner A2                */
       +  (A3:*:B3)*(A2:*:B2)*(A1*/B1)           /* B1 kleiner A1                */
       +  (A3:*:B3)*(A2:*:B2)*(A1:*:B1)*(AO*/BO) /* BO kleiner AO               */
       +  (A3:*:B3)*(A2:*:B2)*(A1:*:B1)*(AO:*:BO) /* B gleich A                 */
```

Bild 5. Ein Beispiel einer Quelldatei mit Simulationsteil für PAL-PROG

funktioniert, wenn bei dem PAL die „Security-Fuse" nicht programmiert wurde, da ansonsten der PAL-Inhalt nicht zugängig ist.

PAL-Simulator

Durch die hohe Komplexität mancher PALs schleichen sich selbst bei erfahrenen Entwicklern leicht Fehler in die Gleichungen ein. Um nun zu überprüfen, ob die im Gleichungsblock vorgegebenen PAL-Terme auch wirklich die gewünschte Funktion definieren, ist der PAL-Simulator vorgesehen. Falls ein PAL simuliert werden soll, muß in der PAL-Quelldatei im Anschluß an die PAL-Terme ein Block folgen, der durch das Wort „SIMULATION" eingeleitet wird. In diesem Block werden als erstes die Anschlüsse definiert, die bei der Simulation berücksichtigt werden sollen. Dann folgen Zeilen, in denen für die vorher definierten Anschlüsse logische Zustände angegeben werden. Diese Zeilen können als eine Art Funktionstabelle aufgefaßt werden. Bei Anschlüssen, die Eingänge darstellen, sind folgende Zustände erlaubt:

„0" : Low-Pegel am Eingang.
„1" : High-Pegel am Eingang.
„X" : Beliebiger Pegel an diesem Eingang.

An den bei Register-PALs vorhandenen Takteingängen sind zwei zusätzliche Angaben erlaubt:
„C" : Taktimpuls (Pegelwechsel von Low nach High).

```
GE =     (/A3*B3)                                  /* B3 größer  A3              */
       + (A3:*:B3)*(/A2*B2)                        /* B2 größer  A2              */
       + (A3:*:B3)*(A2:*:B2)*(/A1*B1)              /* B1 größer  A1              */
       + (A3:*:B3)*(A2:*:B2)*(A1:*:B1)*(/A0*B0)    /* B0 größer  A0              */
       + (A3:*:B3)*(A2:*:B2)*(A1:*:B1)*(A0:*:B0)   /* B gleich A                 */
SIMULATION

TABELLE
 CLK, /OC, CLR, I2, I1, I0, B3, B2, B1, B0, /GE, /LE, /LIO, /RIO, A3, A2, A1, A0

VEKTOREN
/*                 Instr  Eingang                          Ausgang */
/* -Steuerung-    I I I   B B B B   -Status-   -- I/O --   A A A A */
/* CLK /OC CLR    2 1 0   3 2 1 0   /GE  /LE   /LIO /RIO   3 2 1 0 */
/*-------------------------------------------------------------------*/
    C   0   1    X X X   X X X X    -     -      -    -    L L L L /* löschen                          */
    C   0   0    0 1 0   X X X X    -     -      0    H    H L L L /* rechts schieben mit H füllen     */
    C   0   0    0 1 0   X X X X    -     -      1    H    L H L L /* rechts schieben mit L füllen     */
    C   0   0    0 0 0   0 1 0 1    L     H      Z    Z    L H L L /* Vergleich B größer als A         */
    C   0   0    0 1 0   X X X X    -     -      1    H    L L H L /* rechts schieben mit L füllen     */
    C   0   0    0 0 0   0 0 1 0    L     L      Z    Z    L L H L /* Vergleich B gleich A             */
    C   0   0    0 1 0   X X X X    -     -      1    L    L L L H /* rechts schieben mit L füllen     */
    C   0   0    0 1 0   X X X X    -     -      1    H    L L L L /* rechts schieben mit L füllen     */
    C   0   0    1 0 1   X X X X    -     -      H    Z    H H H H /* setzen                           */
    C   0   0    0 0 0   X X X X    -     -      Z    Z    H H H H /* halten                           */
    C   0   0    1 0 0   X X X X    -     -      L    1    H H H L /* links schieben mit L füllen      */
    C   0   0    1 0 0   X X X X    -     -      L    0    H H L H /* links schieben mit H füllen      */
    C   0   0    1 0 0   X X X X    -     -      L    0    H L H H /* links schieben mit H füllen      */
    C   0   0    1 0 0   X X X X    -     -      H    0    L H H H /* links schieben mit H füllen      */
    C   0   0    1 0 0   X X X X    -     -      L    0    H H H H /* links schieben mit H füllen      */
    C   0   0    0 0 1   0 0 0 1    -     -      Z    Z    L L L H /* [1] laden                        */
    C   0   0    1 1 0   X X X X    -     -      H    0    L L H L /* inkrementieren                   */
    C   0   0    1 1 1   X X X X    -     -      H    0    L L L H /* dekrementieren                   */
    C   0   0    0 0 1   0 0 1 1    -     -      Z    Z    L L H H /* [3] laden                        */
    C   0   0    1 1 0   X X X X    -     -      H    0    L H L L /* inkrementieren                   */
    C   0   0    1 1 1   X X X X    -     -      H    0    L L H H /* dekrementieren                   */
    C   0   0    0 0 1   0 1 1 1    -     -      Z    Z    L H H H /* [7] laden                        */
    C   0   0    1 1 0   X X X X    -     -      H    0    H L L L /* inkrementieren                   */
    C   0   0    1 1 1   X X X X    -     -      H    0    L H H H /* dekrementieren                   */
    C   0   0    0 0 1   1 1 1 1    -     -      Z    Z    H H H H /* [F] laden                        */
    C   0   0    0 0 1   0 1 1 1    -     -      Z    Z    L H H H /* [7] laden                        */
    C   0   0    0 0 1   1 0 1 1    -     -      Z    Z    H L H H /* [B] laden                        */
    C   0   0    0 0 1   1 1 0 1    -     -      Z    Z    H H L H /* [D] laden                        */
    C   0   0    0 0 1   1 1 1 0    -     -      Z    Z    H H H L /* [E] laden                        */
    C   0   0    0 0 1   1 0 1 0    -     -      Z    Z    H L H L /* [A] laden                        */
    C   0   0    1 1 0   X X X X    -     -      H    0    H L H H /* inkrementieren                   */
    C   0   0    1 1 0   X X X X    -     -      H    0    H H L L /* inkrementieren                   */
    C   0   0    0 0 0   1 0 1 1    H     L      Z    Z    H H L L /* Vergleich B kleiner A            */
    C   0   0    1 1 0   X X X X    -     -      H    0    H H L H /* inkrementieren                   */
    C   0   0    1 1 0   X X X X    -     -      H    1    H H L H /* halten                           */
    C   0   0    1 1 0   X X X X    -     -      H    0    H H H L /* inkrementieren                   */
    C   0   0    1 1 0   X X X X    -     -      L    0    H H H H /* inkrementieren zu [F]            */
    C   0   0    1 1 0   X X X X    -     -      H    0    L L L L /* inkrementieren zu [0] Überl.     */
    C   0   0    0 1 1   X X X X    -     -      Z    Z    H H H H /* setzen                           */
    C   0   0    1 1 1   X X X X    -     -      H    0    H H H L /* dekrementieren zu [E]            */
    C   0   0    0 0 1   0 0 1 1    -     -      Z    Z    L L H H /* [3] laden                        */
    C   0   0    1 1 1   X X X X    -     -      H    0    L L H L /* dekrementieren zu [2]            */
    C   0   0    1 1 1   0 0 1 0    -     -      H    1    L L H L /* Vergleich B gleich A             */
    C   0   0    1 1 1   X X X X    -     -      H    0    L L L H /* dekrementieren zu [1]            */
    C   0   0    1 1 1   X X X X    -     -      L    0    L L L L /* dekrementieren zu [0]            */
    C   0   0    1 1 1   X X X X    -     -      H    0    H H H H /* dekrementiere zu [F] Überl.      */
    X   1   X    X X X   X X X X    -     -      -    -    Z Z Z Z /* Tristate testen                  */
```

„S" : Diese Zustandsangabe wird benutzt, um Registerausgänge auf einen definierten Pegel zu setzen.

Die für Ausgänge vorgesehenen Zustände sind:

„L" : An diesem Ausgang wird ein Low-Pegel erwartet.

„H" : Hier wird ein High-Pegel erwartet.

„Z" : Dieser Ausgang soll den Tristate-Zustand aufweisen.

„-" : Der Zustand dieses Ausgangs braucht nicht beachtet zu werden.

Der PAL-Simulator berechnet nun, ausgehend von den in den Simulationszeilen vorgegebenen Zuständen, die aus den PAL-Termen folgenden Ausgangszustände. Diese werden dann mit den nach der Tabelle erwünschten verglichen. Stimmen die Zustände nicht überein, so wird eine entsprechende Fehlermeldung ausgegeben.

Um die Bedienung der Module möglichst einheitlich zu gestalten, wurden die PAL-PROG-Befehle denen von EPROM-PROG und PROM-Prog angeglichen. Deswegen sind PAL-Assembler und PAL-Disassembler nicht explizit im Menü aufgeführt. Stattdessen werden sie nur zur Konvertierung Text <-> Sicherungsliste benutzt, und immer dann aufgerufen, wenn eine Funktion eine bestimmte Darstellungsart benötigt, diese aber noch nicht vorhanden ist. Neben den schon bei den anderen Modulen aufgeführten Funktionen findet man unter PAL-PROG noch einige Funktionen, wie zum Beipiel den Simulator oder Hilfsprogramme, um die Sicherungsliste direkt bearbeiten zu können. Selbstverständlich wird auch das Brennen der „Security-Fuse" ermöglicht, was dazu dient, das PAL vor dem Auslesen zu schützen.

Das Listing eines PALs, in dem nahezu alle Möglichkeiten PAL-PROGs benutzt werden, findet sich in *Bild 5*. Weiterhin zeigt dieses Listing recht eindrucksvoll, was so ein 20poliges PAL zu leisten vermag. Abschließend sollte noch gesagt werden, daß die Liste der programmierbaren PALs und PROMs leider nicht so umfangreich ausfallen konnte, wie ursprünglich geplant. Schuld daran tragen die Herstellerfirmen, die sich wenig aufgeschlossen zeigten, Informationen über die Programmier- und Adressierungsmethoden herauszugeben.

Lutz J. Koch

Universal-Interface

1. Teil: Anpassungskarte für den Apple II

Das Universal-Interface besteht aus einer Steckkarte mit Adaptern zum Anschluß an den jeweiligen Rechner, einer Interface-Box mit Stromversorgung zur Aufnahme der Schnittstellen-Karten und den Schnittstellen-Karten selbst. Das Gerät wurde an einem Apple II, einem Commodore C-64 (6502-CPU) und an einem Sharp MZ-700 (Z80-CPU) getestet.

Es war das Ziel bei der Entwicklung des Universal-Interfaces, LSI-Schaltkreise der 6502- und der Z80-Familie gleichzeitig zu verwenden. Um einen Selbstbau zu erleichtern, wurde ein 31poliger Normstecker als Bus-Verbinder verwendet, so daß die von der Zeitschrift Elektor in den Heften Mai/85 und Okt./85 veröffentlichten Schnittstellen ebenfalls verwendet werden können. Diese haben einen 21poligen Normstecker, der in die oberen Pole der von mir verwendeten Buchsen eingesteckt wird.
Folgende Schnittstellen werden beschrieben:

- Eine Parallelschnittstelle mit 4 × 8 Bit. Jeder der vier Kanäle kann als Eingang oder Ausgang geschaltet werden. Bei der Verwendung als Ausgang ist die Pufferung mit 100-mA-Treibern (offener Kollektor) möglich, bei der Verwendung als Eingang sind Pull-up-/-down-Widerstände oder zur Potentialtrennung Optokoppler vorgesehen. Die Platine ist für beide Varianten vorbereitet.
- Eine Parallelschnittstelle mit vier 6522-VIAs, so daß hier 64 Bit mit Handshake-Funktionen und Zeitgebern zur Verfügung stehen. Ein VIA-Baustein wurde für die Centronics-Druckerschnittstelle vorgesehen.
- Eine Parallelschnittstelle mit drei 8255-PIOs mit je 24 Bit, die preiswert und universell verwendbar sind.
- Eine serielle Schnittstelle mit vier 6551-USARTs, die drei voneinander unabhängige V.24-Normschnittstellen sowie eine Midi-Schnittstelle zur Verfügung stellen.

Bei vollem Ausbau können bis zu 256 Byte von dem Universal-Interface belegt werden. Die einzelnen Baugruppen werden bei 6502-Computern als Speicherplätze und bei Z80-Computern im E/A-Bereich angesprochen.

Dabei gelten folgende Adreßbereiche:

Apple II: von Cn00H...CnFFH mit n als Slot-Nummer
C-64: von DE00H...DEFFH
MZ-700: von 00H...EFH, also fast im ganzen E/A-Bereich

Die Anpassungskarte

Diese Steckkarte ist speziell für Apple-Slots vorgesehen. Wird diese mit kleinen Adaptern versehen, so kann sie auch für die anderen erwähnten Rechner verwendet werden. Das in *Bild 1* gezeig-

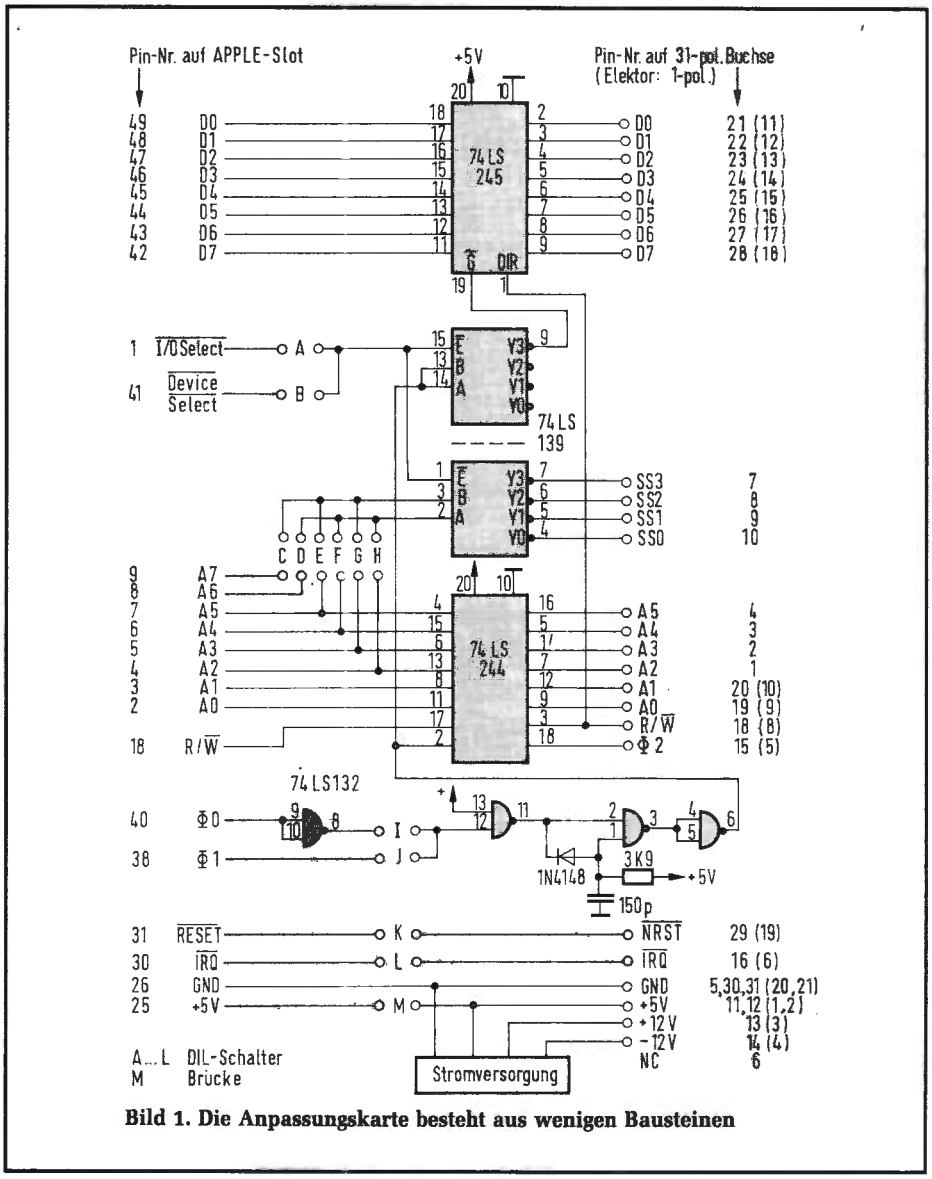


Bild 1. Die Anpassungskarte besteht aus wenigen Bausteinen

te Schaltbild soll die Funktion verdeutlichen. Es sind dazu die Pin-Nummern und -Bezeichnungen des 50poligen Apple-Slots und die des 31poligen Verbinders nach DIN 41617 angegeben. Bei Kombination mit Elektor-Bausätzen ist der dort verwendete 21polige Stecker in die Kontakte 11...31 einzusetzen.

Funktionsweise:

Die Stromversorgung aller Karten wird von einem Netzgerät vorgenommen, das in der Interface-Box eingebaut ist.
Der Datenbus ist mit einem Treiber 74LS245 gepuffert, dessen Richtung vom R/W-Signal bestimmt wird. Er schaltet nur durch, wenn das Φ2-Signal auf High liegt und der Adreßbereich durch die Schalter A und H eingestellt ist.
Ebenso werden die Slot-Select-Signale $\overline{SS0}$...$\overline{SS3}$ nur dann erzeugt, wenn ein E/A-Zugriff erfolgt und die Adressierung richtig ist. Diese Steuerungen werden alle mit dem zweifachen 1-aus-4-Decoder 74LS139 vorgenommen.
Der unidirektionale Bustreiber 74LS244 puffert die übrigen Signale. Ungepuffert bleiben die Leitungen $\overline{IRQ}$ und $\overline{RESET}$, denn hier handelt es sich um durch Pull-up-Widerstände auf 5 V gezogene Ein-/Ausgänge.
Will man die LSI-Interface-Schaltungen der 6502-Serie verwenden, so müssen an diesen Bausteinen alle Steuersignale ca. 150 ns stabil anliegen, bevor die positive Flanke des Φ2-Signals auftritt. Während der positiven Flanke des Φ2-Signals müssen die Steuersignale unverändert anstehen.
Beim Apple ist nun leider das $\overline{I/O\text{-}Select}$-Signal mit dem Φ2-Signal verknüpft, so daß beide Signale nur gleichzeitig auftreten. Deshalb ist der o. g. „Vorhaltezeitraum" von 150 ns hier nicht gegeben. Will man diesen „Fehler" ausgleichen, so besteht eine Lösung darin, die positive Flanke des Φ2-Signals um ca. 150 ns zu verzögern. Dann wird zwar das positive Φ2-Signal kürzer, aber zum Ausgleich kann man dann bei den Schnittstellen-Bausteinen 2-MHz-Typen verwenden, die mit entsprechend verkürzten Takten auskommen. *Bild 2* zeigt die Impulsdiagramme der entsprechenden Signale. Vier NAND-Schmitt-Trigger (74LS132) werden zum Aufbau der Verzögerungsschaltung verwendet.
Wenn das Interface mit einem Z80-Rechner verbunden wird, ist die *Brücke M* erforderlich, um den vorgeschalteten Adapter zu versorgen. Bei dem Betrieb am Apple oder C-64 darf M auf keinen Fall eingelötet sein.
Der *Schalter K* sollte geschlossen werden, da manche der verwendeten ICs am Beginn ein Reset-Signal benötigen. Der

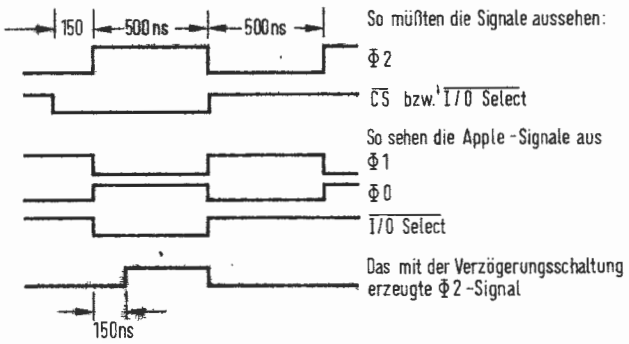


Bild 2. Das Φ2-Signal muß verzögert werden

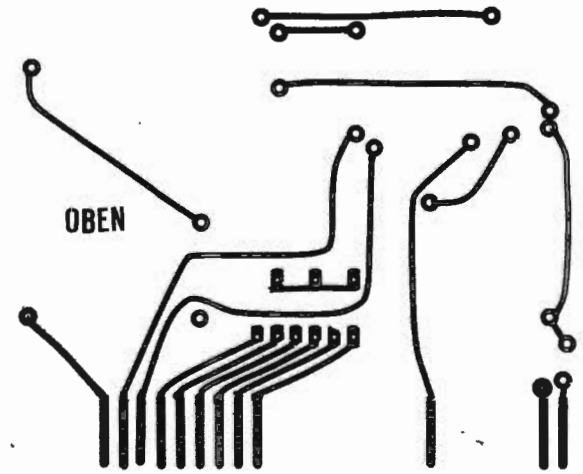


Bild 3. Die Bestückungsseite der Anpassungskarte

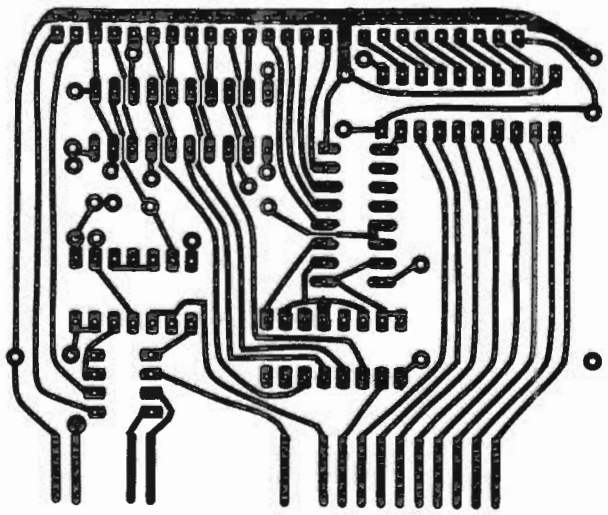


Bild 4. Die Lötseite

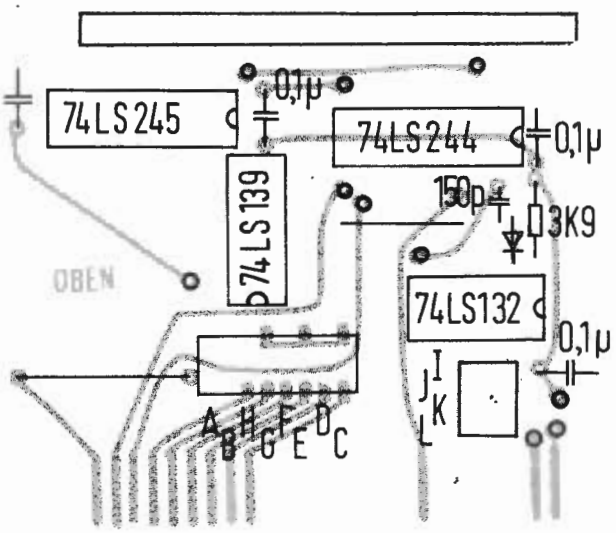
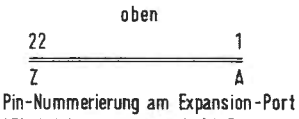


Bild 5. So wird die Platine bestückt

Schalter L sollte nur dann geschlossen werden, wenn man die Interrupt-Struktur des Rechners vollständig beherrscht. Die *Schalter I und J* dienen zur Auswahl des Taktsignales. Beim Betrieb am Apple sollte J geschlossen und I offen sein. Beim C-64-Betrieb dagegen muß I geschlossen und J offen sein. Die nachfolgende Verzögerungsschaltung bewirkt, daß den angeschlossenen Bausteinen der 6502-Serie (2-MHz-Typen) ein ihrem Timing entsprechendes Synchronisationssignal (Φ2) zur Verfügung steht. Auf der Anpassungskarte wird dieses Signal auch zur Steuerung des Adreßdecoders und des Datenbus-Treibers verwendet. So werden Buskollisionen sicher vermieden.
Die *Schalter A bis H* dienen zur Einstellung der Adreß-Decodierung der angeschlossenen Interface-Karten.



Pin-Nummerierung am Expansion-Port
(Blickrichtung auf die C-64-Rückseite)

Bild 6. Blick auf den Erweiterungs-Port des C-64

Z80-Rechner

Hier steht für die Peripherie ein Bereich von 256 Byte zur Verfügung, der mit den Leitungen A0...A7 adressiert und mit dem Signal $\overline{\text{I/O Request}}$ aufgerufen wird. Demgemäß wird dieses Signal mit $\overline{\text{I/O-Select}}$ des Apple-Slots verbunden und der Schalter A geschlossen; Schalter B bleibt offen. Da es sich bei dem IC 74LS139 um einen 1-aus-4-Decoder handelt, ergeben sich nun drei Möglichkeiten:

- Schalter C und D geschlossen, E, F, G, H offen:
  Ergibt vier Adreßbereiche von je 64 Byte, deren jeweilige Aktivierung durch die Signale $\overline{\text{SS0}}$...$\overline{\text{SS3}}$ angezeigt wird.
- Schalter E und F geschlossen, C, D, G, H offen:
  Ergibt vier Adreßbereiche von je 16 Byte, deren Aktivierung durch die Signale $\overline{\text{SS0}}$...$\overline{\text{SS3}}$ angezeigt wird. Diese Bereiche wiederholen sich zyklisch (vierfach gespiegelt).
- Schalter G und H geschlossen, C, D, E, F offen:

Verdrahtungsschema:

Expansion-Port des C-64		APPLE-Slot	
Pin-Nr.	Bez.	Bez.	Pin-Nr.
1	GND	GND	26
4	$\overline{\text{IRQ}}$	$\overline{\text{IRQ}}$	30
5	$\overline{\text{R/W}}$	$\overline{\text{R/W}}$	18
7	I 01	$\overline{\text{I/O Select}}$	1
14	D7	D7	42
15	D6	D6	43
16	D5	D5	44
17	D4	D4	45
18	D3	D3	46
19	D2	D2	47
20	D1	D1	48
21	D0	D0	49
22	GND	GND	26
A	GND	GND	26
C	$\overline{\text{RESET}}$	$\overline{\text{RES}}$	31
E	Φ2	Φ0	40
R	A7	A7	9
S	A6	A6	8
T	A5	A5	7
U	A4	A4	6
V	A3	A3	5
W	A2	A2	4
X	A1	A1	3
Y	A0	A0	2
Z	GND	GND	26

Bild 7. Adapterschema für den C-64

Verdrahtungsschema:

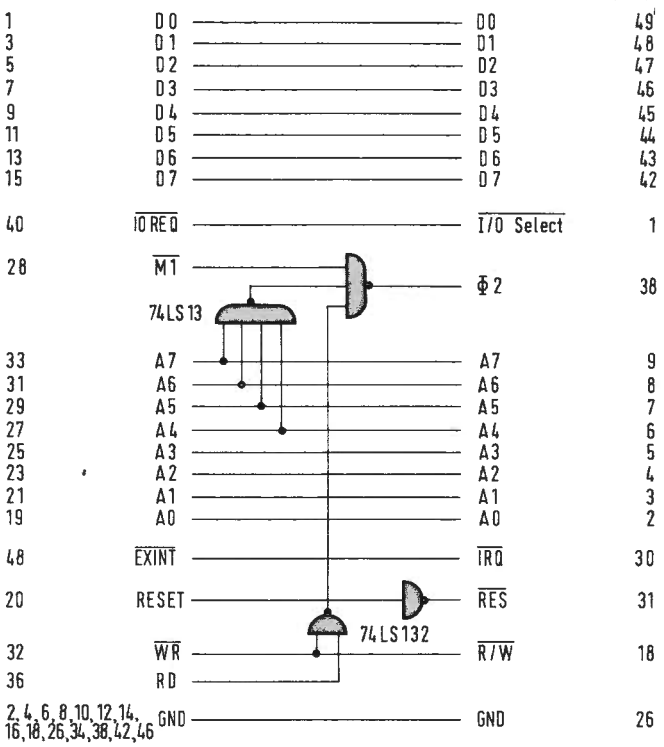


Bild 8. Adapterschaltung für den MZ-700

– Ergibt vier Adreßbereiche von je 4 Byte, deren Aktivierung durch $\overline{SS0}$...$\overline{SS3}$ angezeigt wird. Diese Bereiche wiederholen sich zyklisch (16fach gespiegelt).

Commodore C-64

Hier steht ebenfalls ein 256 Byte breiter E/A-Bereich von DE00H...DEFFH zur Verfügung, dessen Ansprache durch das Signal IO1 (Pin 7) des Expansion-Slots angezeigt wird. Dieses Signal ist an den Pin 1 des Apple-Slots zu legen. Die Adressen der Interface-Karten entsprechen vollständig dem im Z80-Abschnitt beschriebenen Sachverhalt. Die Karten können dann unter Basic einfach mit Poke und Peek angesprochen werden.

Apple II

Hier sind jedem Slot zwei E/A-Bereiche zugeordnet:
– Ein Bereich von 256 Byte von Cn00H...CnFFH, wobei n die Nummer des jeweiligen Slots ist. Wird eine dieser Adressen angesprochen, so geht Pin 1 ($\overline{I/O\ Select}$) auf Low. Bei Verwendung dieses Bereiches ist auf der Anpassungskarte der Schalter A zu schließen. Für die Aufteilung dieses 256-Byte-Bereiches mittels der Schalter C...H gilt das gleiche wie für den Z80.
– Ein Bereich mit 16 Byte von C0n0H bis C0nFH mit n als Slot-Nummer.

Wird eine dieser Adressen angesprochen, so geht Pin 41 ($\overline{Device\ Select}$) auf Low. Will man also mit diesem Bereich arbeiten, so sind auf der Anpassungskarte die Schalter B, G und H zu schließen. Damit kann der Adreßdecoder die 16 Byte in vier Bereiche von je vier Byte aufteilen und mit $\overline{SS0}$...$\overline{SS3}$ anzeigen. Somit stehen viele Möglichkeiten des mehr oder weniger umfangreichen Ausbaus des Interface-Systems offen. Das Layout der Anpassungskarte und der Bestückungsplan wird in den *Bildern 3* bis 5 gezeigt. Alle Stützkondensatoren haben den Wert 0,1 µF. Die Verbindungen zwischen den beiden Platinenseiten sind durch Punkte markiert, sofern sie nicht durch IC-Fassungen, Stützkondensatoren oder Drahtbrücken erfolgen.

Adapter für C-64 und SHARP MZ-700

Um die Anpassungskarte, außer im Apple, noch in anderen Computern betreiben zu können, hat unser Autor Adapterschaltungen verwendet. Diese bestehen aus einer kleinen Platine, die auf einer Seite mit einem 50poligen Apple-Slot und auf der anderen Seite mit dem entsprechenden Stecker des jeweiligen Computers versehen sind. Damit braucht man nur diesen Adapter auf die Anpassungsplatine zu stecken und kann dann sofort mit dem anderen Computer arbeiten.
Erfreulicherweise werden für den C-64 keine Zusatzschaltungen benötigt. Es genügt, die Anschlüsse so zu verbinden, wie es in den *Bildern 6* und *7* angegeben ist. Für die Adapterkarte des MZ-700 (*Bild 8*) werden ein Vierfach-NAND 74LS13 und ein 74LS132 gebraucht. Das Vierfach-NAND sperrt die Portadressen F0H...FFH für das Interface-System, da hier sharpspezifische Geräte arbeiten. Bei anderen Z80-Systemen könnte dieser Teil entfallen.

Hochintegrierte PC-Bausteine

Der kalifornische Halbleiterhersteller Zymos erweitert seine POACH-Familie (PC ON A CHIP). Die wichtigsten Komponenten eines PCs sind in dem Baustein POACH/XT88 zusammengefaßt. Dazu gehören der Interrupt-Controller 8259, der Zeitgeber 8254, der Taktgenerator 8284A, der Bus-Controller 8288, der DMA-Controller 8237A und der Parallel-Ein-/-Ausgabebaustein 8255. Die meisten der restlichen Bausteine auf der PC-Systemplatine ersetzt der Baustein POACH/XTB. Dem Stand der Technik entsprechend werden diese Bausteine in der CHMOS-Technologie hergestellt. In dieser Technik wird der sparsame Stromverbrauch der CMOS-Technik mit der höheren Geschwindigkeit der HMOS-Technik vereint.
Auch für PCs der AT-Klasse gibt es von Zymos einen entsprechenden Chipsatz. *Bild 1* stellt das Blockdiagramm eines ATs dar, der mit den POACH-Bausteinen bestückt ist. Im Baustein POACH 1 sind der Taktgenerator 82284, der Bus-Controller 82288, zwei Interrupt-Controller 8259, die Echtzeituhr 6818, die Schnittstelle zum Arithmetik-Prozessor 80287 und noch einige zusätzliche Logikfunktionen untergebracht. Weitere Standard-AT-Bausteine sind in POACH 2 integriert. Dazu gehören der Zeitgeber 8254, zwei DMA-Contoller 8237, das DMA-Seiten-Register 74612, der DRAM-Refresh-Zähler, die Logik zum Erkennen von Paritätsfehlern und verschiedene weitere Schnittstellen. Alle Bausteine sind bereits für die Oberflächenmontagetechnik konzipiert.
Der Entwicklung eines PCs oder eines ATs im Europakarten-Format steht damit nichts mehr im Wege.

Nach Unterlagen von Zymos, Sunnyvale

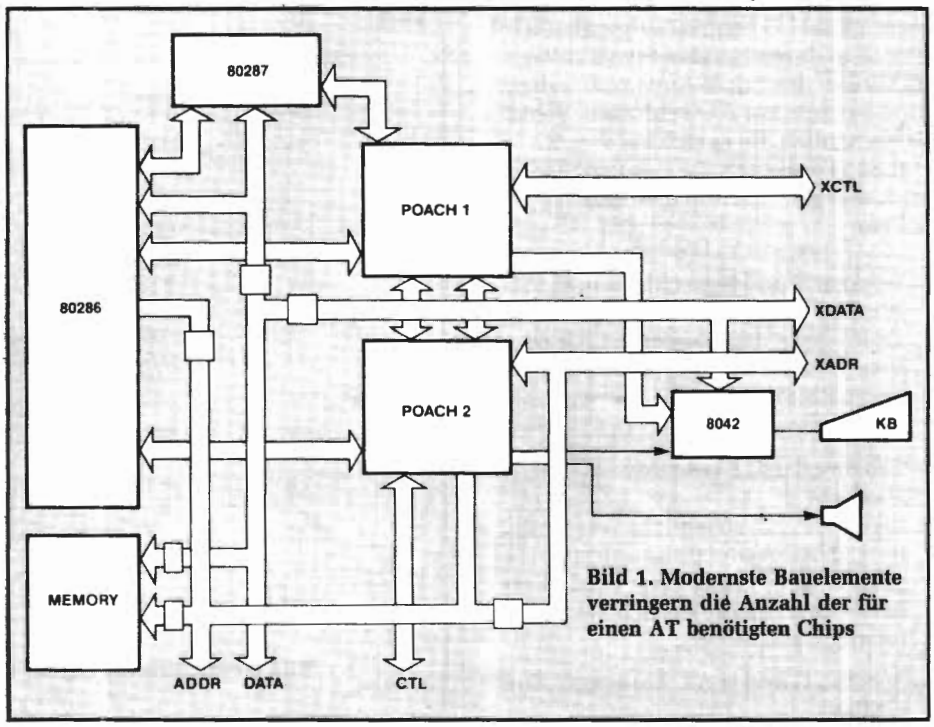


Bild 1. Modernste Bauelemente verringern die Anzahl der für einen AT benötigten Chips

Jürgen Plate

C mit Umgebung

Test: Turbo-C von Heimsoeth/Borland

Nun gibt es ihn endlich – den schon lange erwarteten C-Compiler aus der Turbo-Pascal-Schmiede. Nach Pascal, Prolog und Basic waren die Erwartungen hoch gesteckt, die an einen C-Compiler gestellt wurden. Turbo-C im ersten Test.

C ist die Programmiersprache, die zur Zeit von Entwicklern systemnaher Programme mit Vorliebe verwendet wird. Mir lief es zunächst einmal kalt über den Rücken, als ich einen C-Compiler testen sollte, denn ich bin ein alter Pascal-Freak, und meine bisherigen C-Tätigkeiten waren von dem umständlichen Ablauf von Editieren, Übersetzen, Linken, ... bei den klassischen C-Compilern überschattet.

Das ist natürlich bei Turbo-C auf die übliche Borland-Manier anders geworden. Alle Funktionen sind beim Turbo-C-System über „Rolladenmenüs" innerhalb des C-Systems erreichbar. Bei Borland-Produkten eine Selbstverständlichkeit, über die hier nicht weiter gesprochen werden soll. In *Bild 1* sind alle Menüs zusammengefaßt zu sehen (ich hatte zum Test noch die amerikanische Version, ein deutsches C ist in Kürze fertig). Interessant ist die Funktion „OS Shell", mit der man zwischendurch ins Betriebssystem springen kann. Über das Project-Menü kann man größere Programmierprojekte verwalten. Das Arbeitsfenster ist zweigeteilt in ein Editor- und Meldungsfenster. Man kann mit einer Zoom-Funktion bestimmte Fenster auf den gesamten Bildschirm ausdehnen. Die Behandlung von Compiler-Fehlern ist anders als bei Turbo-Pascal. Alle Fehler werden sofort ermittelt , wobei nicht beim ersten Fehler angehalten wird. In einem Message-Fenster werden die Fehler angezeigt. Gleichzeitig wird die entsprechende Quellzeile im Editor farbig unterlegt. Man kann sich nun alle Fehler in Ruhe ansehen. Beim Wechsel in den Editor kann man nun zur Korrektur von Fehlerposition zu Fehlerposition springen – eine feine Sache. Eine Hilfsfunktion unterstützt den Einsteiger.

Die Codeerzeugung

Im Gegensatz zu Turbo-Pascal erzeugt der C-Compiler einzelne Programm-Module, die dann zusammengelinkt werden können. Normalerweise wird der Linker automatisch nach dem Compilieren aufgerufen, kann aber auch als eigenständiges Programm verwendet werden. Auf diese Weise lassen sich C-Programmteile in Assemblerprogramme einbinden. Natürlich fehlt auch das von Unix her bekannte Make Utility nicht. Turbo-C erzeugt EXE-Dateien beliebiger Größe, daher sind auch keine Overlays vorgesehen. Es gibt verschiedene Speichermodelle (Tiny = 64 KByte für Speicher und Code zusammen; Small =

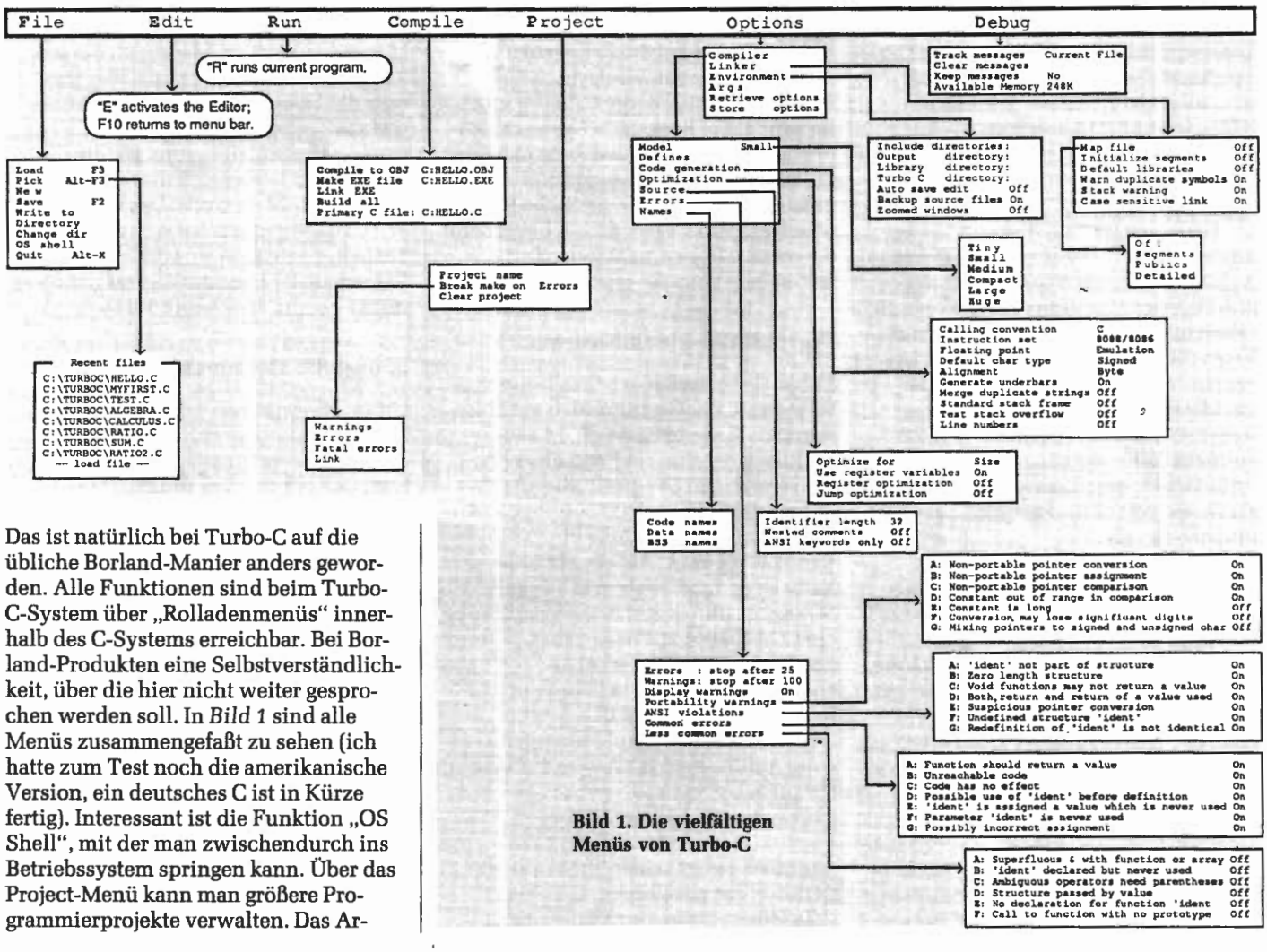


Bild 1. Die vielfältigen Menüs von Turbo-C

64 KByte für Code und 64 KByte für Daten; Medium = Verwendung von Far-Pointern im Code, 64 KByte für Daten; Compact = 64 KByte für Code und Far-Pointer für Daten; Large = Far-Pointer für Code und Daten mit der Ausnahme „static"; Huge = Far-Pointer auch für static). Zu Torbo-C gehört noch ein zweiter Compiler, der nicht über Menüs, sondern konventionell über eine Kommandozeile angesteuert wird. Da der Linker auch als eigenständiges Programm verfügbar ist, kann man also auch über Batch-Dateien (oder Make) komplette Übersetzungsläufe starten. Nach erfolgreicher Übersetzung wird eine kurze Statistik ausgegeben. Der C-Compiler ist ein schneller Einpaß-Compiler (nach Herstellerangaben 7000 Zeilen/min auf IBM-AT). War die Übersetzung fehlerfrei, folgt als zweiter Durchlauf sofort der Link-Vorgang. Über den eingebauten Lint (unter Unix-Programmierern bekannter Syntax- und Semantik-Prüfer) kann die Syntax und die Konsistenz des Quellprogramms ohne das Erzeugen von OBJ- und EXE-Dateien überprüft werden. Nach den ersten Eindrücken scheint der Compiler zudem einen sehr kompakten Code zu erzeugen, der für die Prozessoren 8086 ... 80286 zu optimieren ist.

Der Sprachumfang

Der Sprachumfang läßt keinen Wunsch offen. Er folgt dem neuen ANSI-Vorschlag, also auch dem „Standard" von Kernighan und Ritchie. Doch auch die MS-DOS-Seite wird von Turbo-C gut unterstützt: Assembleranweisungen lassen sich nicht nur als Maschinencode, sondern auch als ganz normale Assembler-Mnemonics beliebig in den C-Quelltext einstreuen. Dabei kann direkt auf C-Variablen zugegriffen werden. Für die Register sind Pseudo-Variablen vordefiniert. Auf diese Angebote muß man aber meist überhaupt nicht eingehen (denn sie sind nicht einfach zu nutzen), weil eine Riesenbibliothek von DOS-Funktionen bereits vorhanden ist. So gibt es über 300 Funktionen zum Schreiben von Interrupt-Routinen, für den Zugriff auf die BIOS- und BDOS-Funktionen, Prozeß-Management und die IEEE-Gleitkommaarithmetik. Wie einfach es ist, systemnah zu programmieren, soll das Beispiel in *Bild 3* zeigen: Ich wollte ein kleines Programm zur Ausgabe des Directory schreiben, da hier ein direkter Vergleich mit dem Aufwand unter Turbo-Pascal möglich ist (ein entsprechendes Programm findet sich z. B. im Sonderheft „Turbo-Pascal im Einsatz"). Für den File-Control-Block ist ein Record vordefiniert, den ich im Programm als Kommentar mit aufgenommen habe. Mit der vordefinierten Funktion „findfirst" wird der erste Eintrag im Directory gesucht. Der erste Parameter gibt den Pfad an, der hier das ganze, aktuelle Verzeichnis umfaßt. Der zweite Parameter enthält die Adresse des FCB und der dritte Parameter wählt das Attribut (0 für die „normalen" Dateien) aus. Wurde ein Eintrag gefunden liefert die Funktion den Wert 0. Weiter gesucht wird mit der Funktion „findnext", die nur noch den FCB als Parameter hat. Sie wird sooft aufgerufen, bis die Variable „done" den Wert -1 annimmt. Sie sehen, wie kurz und schnell das Ganze programmiert werden kann. Das fertige EXE-File ist 5,5 KByte lang.

Das Umfeld

Turbo-C benötigt mindestens 320 KByte RAM. Das ist aber immer noch sehr knapp. Besser sind 512 bis 640 KByte. Durch die große Zahl von Include- und Lib-Dateien (das Paket besteht aus vier 360-KByte-Disketten) ist für komfortables Arbeiten eine große RAM-Disk oder eine Festplatte zu empfehlen. Das bedeutet aber nicht, daß der Compiler nicht auch nur mit ein oder zwei Diskettenlaufwerken verwendet werden kann – man wird dann nur leicht zum „Diskjokey". Mit dem Compiler kommen zwei Handbücher, deren Qualität dem „Borland-Standard" entspricht: ein dicker „Reference Guide", in dem alle Standard-Variablen und -Funktionen beschrieben sind und ein „Users Guide", der den Einstieg in Turbo-C beschreibt.

Das beginnt mit dem Installieren der Programme und der Beschreibung der Menüs. Dann folgt eine Einführung in die Programmierung. Es schließen sich zwei Kapitel für Umsteiger von Pascal und Turbo-Prolog an. Hier werden nicht nur Hinweise gegeben, sondern auch die möglichen Fallstricke exemplarisch abgehandelt. Das Handbuch endet mit einem Referenzteil, der auch auf die Erweiterungen gegenüber K&R eingeht und einem Kapitel über fortgeschrittene Programmiertechniken. Wer noch nie in C programmiert hat, sollte sich noch die „Bibel" von Kernighan und Ritchie dazu besorgen.

Resümee

Mit Turbo-C ist ausgesprochen angenehm und effizient zu arbeiten – und über den Preis von knapp 400 DM kann man wirklich nicht meckern. Und für alle, die es ganz genau wissen wollen, bietet Borland zusätzlich den Quellcode (!) der Runtime-Library an (USA-Preis 295 $).

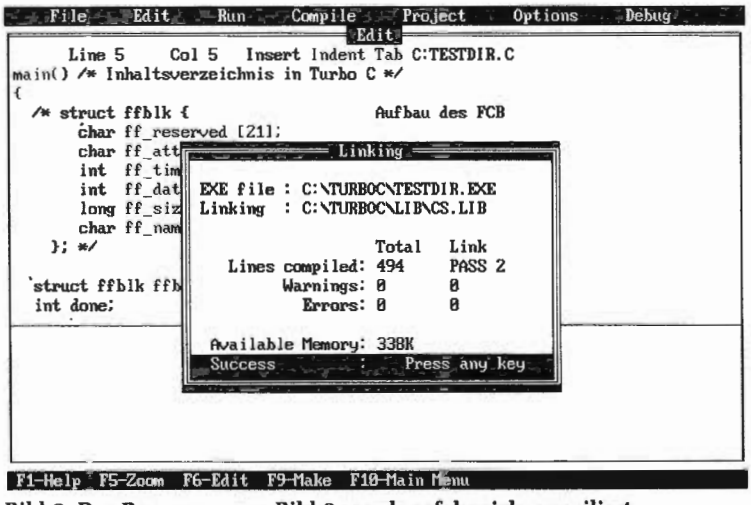


Bild 2. Das Programm aus Bild 3 wurde erfolgreich compiliert

```
#include <stdio.h>
#include <dos.h>
#include <dir.h>

main() /* Inhaltsverzeichnis in Turbo C */
{
  /* struct ffblk {                    Aufbau des FCB
       char ff_reserved [21];
       char ff_attib;
       int  ff_time;
       int  ff_date;
       long ff_size;
       char ff_name[13];
     }; */

  struct ffblk ffblk;
  int done;

  printf("Directory of *.*\n");
  /* Suchen ersten Eintrag */
  done = findfirst("*.*",&ffblk,0);
  while (!done)
    {
    printf("  %s\n",ffblk.ff_name);
    done = findnext(&ffblk);
    }
  }
```

Bild 3. So einfach ist das Auflisten des Directory

Dieter Strauß

Exklusiv für Profis

QNX: Ein Echtzeit-Betriebssystem für den PC

Programmierer, die ein Unix-ähnliches Betriebssystem suchen, sind mit QNX von Quantum gut bedient. Im Gegensatz zu vielen Unix-Derivaten ist QNX „schlank", d. h. es belegt wenig Speicherplatz. Seine Fähigkeit zur Echtzeitverarbeitung macht es ideal für Anwendungen in Netzwerken und zur Prozeßautomation.

Hierzulande ist QNX noch wenig bekannt, aber dennoch ist es weltweit auf bereits über 30 000 Computern der PC- und der AT-Klasse installiert. In Deutschland wird QNX von repas, einem Softwarehaus mit Sitz in Dreieich, vertrieben.
mc hat die QNX-Version, die den „protected mode" des 80286-Prozessors anspricht, auf einem MCI AT 4SLC getestet. Neben dieser Version gibt es noch eine netzwerkfähige und eine PC-Version.
QNX ist kein alternatives Betriebssystem zu MS-DOS für Anwender, die unter MS-DOS Kalkulationsprogramme, Datenbanken, Turbo-Pascal-Anwendungen, Textverarbeitung, Desktop Publishing und dergleichen durchführen. Diese Anwender werden auch in Zukunft mit MS-DOS und Microsofts neuem Betriebssystem (OS/2) besser bedient sein. Für technische Anwendungen, insbesondere der Prozeßautomation, oder für Netzwerk-Server ist QNX jedoch zweifellos die bessere Wahl.
QNX bietet vieles, was MS-DOS nicht kann:

- Mehrbenutzerfähig: Bis zu zehn Terminals pro PC können bedient werden.
- Multitasking: Bis zu vierzig Tasks können quasi simultan ablaufen.
- Echtzeitverarbeitung: Ein deterministischer Schedule-Algorithmus sorgt dafür, daß auf wichtige Ereignisse innerhalb einer bestimmbaren Zeit reagiert wird. Bei einem AT ist die Antwortzeit weit unter 1 ms.

QNX braucht wenig Speicherplatz

Um zu prüfen, ob QNX auch auf einem abgespeckten AT läuft, haben wir den MCI AT4SLC mit nur 512 KByte RAM betrieben. Bereits nach dem Booten stellt man fest, daß man sich mit seinem System in einer ganz anderen Welt befindet – der Welt der Unix-ähnlichen Betriebssysteme. Leser unserer Serie „Einführung in Unix" werden sehr viele der dort beschriebenen Befehle in *Bild 1* wiederfinden. Im Gegensatz zu Unix und dessen zahlreichen Derivaten, braucht QNX keinen Computer mit einem riesigen Speicher. Für kleinere Anwendungen reichen 512 KByte bereits völlig aus. Auch eine Festplatte ist nicht unbedingt nötig. Aus eigener Erfahrung weiß ich jedoch, daß jeder, der einmal mit einer Festplatte gearbeitet hat, auf diese nie mehr verzichten möchte. Auch bei QNX ist daher eine Festplatte – zumindest zur Programmentwicklung – empfehlenswert.

Im QNX-Handbuch, das leider nur in Englisch erhältlich ist, werden die ersten Schritte mit diesem Betriebssystem beschrieben. Benutzer ohne Unix-Kenntnisse werden sich dennoch schwer tun. Bereits der Befehl zum Auflisten des Directories lautet völlig anders als es MS-DOS-Anwender gewöhnt sind. Das Kommando „ls 1:/" gibt das Wurzel-Verzeichnis der Diskette in Laufwerk A auf den Bildschirm aus.
Etwas unübersichtlich ist die Installierung einer Festplatte dargestellt. Auf über sechzehn Seiten ist zwar alles wesentliche beschrieben, ohne diese jedoch sehr sorgfältig zu lesen, wird man bei der Installierung leicht einen Fehler machen.
Es ist ohne weiteres möglich, eine Parti-

Listing directories and files

dump	- Dump a file in hexadecimal.
expl	- Online explain files.
files	- List contents of your current and all lower directories.
list	*- List files on the line printer.
ls	*- List contents of your current directory (names of your files).
p	*- Print a file on your terminal.
pwd	*- Print working directory.

Editing and word processing

ed	*- Full screen editor.
led	- Line editor.

File and directory manipulation

cd	*- Change current directory.
chattr	- Change the attributes (name,permissions) of a file.
drel	- Release directories.
eo	- Execute a command on a file of files.
frel	*- Release files.
mkdir	*- Create a directory.
patch	- Patch data and load files.
rm	- Release files.
rmdir	- Release directories.
ws	*- Walk a directory structure executing a command.
zap	- Remove a corrupted file.

Moving files

arch	- Archive files to a floppy.
backup	*- Backup files to another disk and directory.
cat	- Concatenate several files into one.
copy	*- Copy one file to another.
cp	*- Copy one or more files to another.
split	- Split one file into several.

Languages and Compilers

asm	- Assemble 8088/8087 program and generate an object file.
cc	- Compile program and link it.
cco	- Compile program and generate an object file.
link	- The linker.
linkobj	- Link together a object files into a load file. This invokes LINK with proper arguments.
make	- Make a program.

Disk utilities

chkfsys	- Check the file system integrity.
dinit	*- Initialize a disk.
dcheck	- Check for bad blocks on a disk.
dcopy	*- Make an exact copy of a diskette.
ddump	- Dump disk blocks in hexadecimal.
fdisk	*- Partition fixed disk.
fdformat	*- Format a brand new floppy diskette.
query	*- Disk usage.
spatch	- Patch a disk.

Information Handling

crypt	- Encrypt/decrypt a file.
diff	- Compare files for equality.
locate	*- Locate patterns within files.
msort	- Merge sort.
pack	- Reduce a files size.
size	- Size of a file.
sort	- Sort a file.
wc	- Number chars, words and lines in a file
xlat	- Translate a file

QNX utilities

apb	- Send a message to all users currently logged on.
beep	- Send a message to a user or terminal.
break	*- Break a background task.
cron	- Dispatch tasks at a specified time.
date	*- Get/set system date and time.
kill	- Kill a background task.
mount	*- Mount new disk drivers/memory/debugger.
nacc	- Define network access
net	- Find network statistics
patch	- Patch a specified byte in a file.
passon	- Turn on password protection at Login.
search	*- Define disk search order.
slice	- Set task time slice.
spool	- Spool output to the system spooler.
task	*- List tasks running in the system.
tcap	*- Set terminal type
who	*- Find out who's logged on.

Network QNX utilities

alive	- Up/down status of a node.
kill_vcs	- Kill any virtual circuits to a node.
poll	- Network poller

Communication

comm	- Communication task for dial in modems.
stty	*- Set terminal options.
talk	*- Talk to mainframes and other PC's.

Bild 1. Sehr viele QNX-Befehle stammen aus der Unix-Welt

tion der Festplatte für MS-DOS und den anderen Teil für QNX zu verwenden.

Einfacher Editor

Der Pedant zu CONFIG.SYS unter MS-DOS lautet unter QNX /config/sys.init. Um QNX an das jeweilige System anzupassen, können in dieser Datei Angaben gemacht werden zur Größe der RAM-Disk, der Suchreihenfolge bei verschiedenen Laufwerken, der Baudrate der seriellen Schnittstelle usw. Die Eingaben in /config/sys.init müssen mit dem mitgelieferten Editor namens „ed" durchgeführt werden. Bereits bei diesem Namen zuckte ich zusammen: Augenblicklich fiel mir die umständliche Bedienung des MS-DOS-Editors EDLIN ein. Sofort nach dem Starten von „ed" wird einem jedoch klar, daß man es hier mit einem modernen Editor zu tun hat. Sogar die von Wordstar bekannte Fingerakrobatik bleibt dem Anwender erspart. Die obere Bildschirmzeile dient als Status-Zeile und die nächste Zeile als Kommando-Zeile. Mit den Cursor-Tasten und der Kombination der Alt-Taste mit nur einer weiteren Taste lassen sich alle Funktionen aufrufen.

Treiber einfach installiert

Ein wichtiges Kommando, das bereits in der /config/sys.init-Datei verwendet wird, ist „mount". Mit diesem Kommando werden dem System neue Disketten-Laufwerke, neue Treiber, eine RAM-Disk usw. bekanntgemacht. Das Kommando „mount disk 3 /drivers/at pa=qnx" teilt QNX mit, daß eine Festplatte mit dem Treiber für den AT als drittes Laufwerk (entspricht C bei MS-DOS) angesprochen werden soll. Ohne Argumente gibt „mount" eine Liste aller initialisierten Laufwerke, aller Treiber und aller aktivierten Bibliotheken aus.
Mit „mount" kann der Anwender auch den Debugger oder eine sogenannte „shared library" laden. Eine „shared library" ist eine Bibliothek, in der Routinen stehen, die per Software-Interrupt von allen Tasks aufgerufen werden können. Bibliotheks-Funktionen, wie z. B. Fließkomma-Routinen, stehen damit nur einmal im Speicher.

Multitasking

QNX kann bis zu 40 quasi simultan ablaufende Tasks in insgesamt 15 Prioritätsebenen verwalten.
Nach dem Booten laufen bereits vier System-Tasks: Task Administrator, File System Administrator, Device Administrator und Idle Administrator.
Der Task Administrator führt die Speicherverwaltung durch und erzeugt und vernichtet Tasks. Er hat die höchste Priorität (Priorität 1). Vom File System Administrator wird das Datei-System verwaltet. Alle Zugriffe auf zeichenorientierte Betriebsmittel (Terminal, Drucker) werden vom Device Administrator abgewickelt. Freie Prozessorzeit verbraucht der Idle Administrator. Er läuft mit der geringsten Priorität.

Botschaften austauschen

Zur Kommunikation zwischen den einzelnen Tasks verwendet QNX ein Verfahren, das auf dem Austausch von Botschaften (Message Passing) beruht. Eine Botschaft besteht aus einer Folge von bis zu 65535 Byte. Auch zwischen den Tasks, die auf verschiedenen Knoten eines Netzwerkes arbeiten, können Botschaften ausgetauscht werden. Die Länge der Botschaften zwischen Tasks auf verschiedenen Netzwerk-Knoten ist allerdings durch die Größe des Puffers der virtuellen Verbindung begrenzt.
Der Inhalt einer Botschaft muß einer Struktur entsprechen, die sowohl dem Sender als auch dem Empfänger bekannt sein muß. Es ist die Aufgabe der jeweiligen Task, diese vereinbarte Struktur einzuhalten. Vom Betriebssystem wird die Konsistenz einer Botschaft nicht geprüft. Es betrachtet eine Botschaft als eine Byte-Folge ohne jede Struktur. Übrigens hat QNX die gleiche Sichtweise auch gegenüber Dateien.
Unter QNX wird zwischen sieben verschiedenen Task-Zuständen unterschieden:
Ready: Die Task ist bereit zur Ausführung.
Receive Blocked: Die Task wartet auf den Empfang einer Botschaft. Benötigt eine Task z. B. den Zugriff auf eine Datei, so schickt diese die Botschaft an den File System Administrator: „Öffne die gewünschte Datei". Erst wenn der File System Administrator geantwortet hat, daß die Datei geöffnet ist, kann die jeweilige Task den Zustand „Receive Blocked" verlassen.
Send Blocked: Die Task hat eine Botschaft gesendet, deren Empfang von der Empfänger-Task nocht nicht bestätigt wurde.
Reply Blocked: Die Task hat eine Botschaft abgesendet, die der Empfänger bereits quittiert, jedoch noch nicht beantwortet hat.
Network Blocked: Die Task wartet auf eine Anforderung vom Netzwerk.
Dead: Die Task wird vernichtet oder sie existiert nicht.
Held: Eine Task ist bereit zur Ausführung. Sie wird jedoch in ihrer Ausführung angehalten.

Mit dem Kommmando „task" kann sich der Anwender jederzeit einen Überblick über die derzeit existierenden Tasks verschaffen. *Bild 2* zeigt ein Beispiel für dieses Kommando. In der Antwort des Systems werden z. B in der Rubrik tid die Identitätsnummer (tid = task identity) und in der Rubrik pri die Priorität der jeweiligen Task angegeben.

Dokumentation in C

Gut gefallen hat mir die Dokumentation des Datei-Systems. Wesentliche Eigenschaften wie der Aufbau von Directories oder des Lade-Formats von QNX-Dateien sind im technischen Anhang des Betriebssystems-Handbuchs und des Handbuchs zum C-Compiler bereits als Record-Definition in der C-Syntax beschrieben. Für C-Programmierer dürfte dies eine Augenweide sein. Leider ist im Handbuch nichts über mögliche Tastatur-Anpassungen erwähnt. Auf der Tastatur des AT4SLC waren fast alle Sonderzeichen nur nach einer Suchaktion zu finden.
Die Arbeitsgeschwindgkeit des ATs verlangsamt sich durch das Multitasking nur unwesentlich, solange keine Task mit rechenintensiven Operationen unter einer hohen Priorität läuft.
Insgesamt gesehen ist QNX für den professionellen Anwender und Programmierer eine Alternative zu MS-DOS in speziellen Anwendungen.

```
$ task
name          tid  state    pri  flags     grp usr tty blked father son  brother
task          0001 READY     1  PLA------ 255 255  0  ----  ----  ----  ----
fsys          0002 RECEIVE   3  PLA------ 255 255  0  0000  ----  ----  ----
dev           0003 RECEIVE   2  PLA------ 255 255  0  0000  ----  ----  ----
idle          0004 READY    15  PLA------ 255 255  0  ----  ----  0205  ----
/cmds/sh      0205 REPLY     8  P----E--- 255 255  0  0001  0004  1007  ----
/cmds/clock   0406 READY    15  ---C----- 255 255  0  ----  ----  ----  ----
/cmds/task    1007 REPLY     8  -------- 255 255  0  0001  0205  ----  ----
```

Bild 2. Acht Tasks laufen quasi simultan

Wie? Kompetente Beratung und auch noch günstige Preise?

JA, GENAU: Bei VOBIS! [illegible] wert!

Gebühr bezahlt
0437295/008/87 539

HEWLETT PACKARD RECHNER

11011 HP 11 C programmierbar	119.00
11012 HP 12 C kaufmaennisch	199.00
11015 HP 15 C programmierbar	219.00
11016 HP 16 C f. Programmierer	275.00
11028 HP 28 wissenschaftlich	475.00
11041 HP 41 CV 2233 Schritte	349.00
11042 Thermodrucker fuer 41 C	1098.00
11043 HP 41 CX + Time,X-Funct.	525.00
11047 IL-Bandlaufwerk	1295.00
11048 IL-Drucker	1195.00
11049 HP-IL Modul	369.00
11050 HP-IL-Kabel 0.5 Meter	20.00
11053 X-Memory-Modul	218.00

PHILIPS TASCHENRECHNER

11810 SBC 1745	39.00

SHARP RECHNER U. COMPUTER

12023 RD 720H Datenrecorder	99.00
12034 PC 1403	239.00
12035 PC 1403 erw. auf 32 K Ram	299.00

PREISSENKUNG:

12038 PC 1260 2 reihige Anzeige	179.00
12039 CE 125 Drucker f. 12xx	225.00

PREISSENKUNG:

12040 PC 1260 erweitert auf 10K	199.00

PREISSENKUNG:

12041 CE 126 P Drucker	159.00
12043 CE 129 P Drucker	199.00
12350 PC 1360 8 K	375.00

CASIO RECHNER U. COMPUTER

13020 FX-82 Schulrechner	26.00

PREISSENKUNG:

13400 PB-220 G Basic 10 KB	99.00
13420 FX-730 P Pocket Computer	169.00
13423 RP-8 Erw.auf 16K f.FX-730	45.00
13450 PB 1000 Pocket Computer	375.00
13451 PB 1000 erw. auf 40 K	498.00
13453 RP-32 Erw.auf 40K PB 1000	149.00

PREISSENKUNG:

13460 MD 100 3.5" Floppy Disk	648.00
13462 AD-4175 Netzteil f.MD 100	35.00
13470 FA 7 Cas,V 24,Centr. Intf	199.00
13510 FP 12-s Thermodrucker	99.00
13520 FP-40 40 Zeichen-Drucker	225.00

COMMODORE COMPUTER

PREISSENKUNG:

20021 C 64 Computer 38 K Ram	333.00
20023 VC 1541 Floppy 170 K	399.00

PREISSENKUNG:

20024 C 64 II Computer m.GEOS	395.00

PREISSENKUNG:

20025 VC 1551 Floppy f. C16/+4	279.00

PREISSENKUNG:

20032 PLUS 4 Computer 64 K	178.00
20034 TURBO PLUS f.C16/PLUS 4	49.00

PREISSENKUNG:

20035 GEOS f. C 64/64-II	29.00
20037 Maus f. C 64/64-II/128	79.00
20040 VC 1520 Printer-Plotter	199.00

PREISSENKUNG:

20047 MPS 1000 VC/PC-Interface	648.00
20065 MPS 803 Matrixdrucker	399.00

PREISSENKUNG:

20069 Citizen 2-FarbDrucker C64	79.00

PREISSENKUNG:

20076 FINAL-CARTRIDGE GP 700 A	79.00

PREISSENKUNG:

20087 1802 Farbmonitor	449.00
20100 COMMODORE C 128	548.00
20105 COMMODORE 128 D	1076.00
20112 VC 1571 Floppy f. C 128	569.00
20117 C128-LEARN-Maschinenspr.	98.00
20118 C128-FORTH-Compil.Interp.	98.00
20119 C128-COM-Kommunikation	98.00
20165 SUPERSCRIPT Textprogramm	85.00
20170 SUPERBASE Dateiprogramm	85.00
20330 VC 1530 Datenrecorder	68.00
20332 VC 1531 Datenrecorder	59.00
20342 Joystick-Adapter f. C 16	9.00

AMIGA COMPUTER

20812 AMIGA 500 Basispaket	1235.00
20815 AMIGA 1000 Basispaket	1695.00

PREISSENKUNG:

20817 AMIGA 2000 Basispaket 1MB	2575.00

PREISSENKUNG:

20823 1081 RGBI AMIGA-Monitor	745.00
20831 2. Laufwerk A2000(intern)	348.00
20832 1010 2. AMIGA-Laufwerk	399.00
20841 A 501 Ram-Erweit. F. A500	298.00
20842 A 520 HF-Modulator f.A500	59.00
20853 MPS 2000 C(olor)	1499.00
20855 Traktor f. MPS 2000	298.00
20857 Einzelblatteinzug NEC P 6	599.00
20880 SIDECAR-PC Aufruestung	1695.00

PREISSENKUNG:

20900 PC-Board f. AMIGA 2000	1198.00
20950 Erw. A2000 auf 1.5 MB	249.00

APPLE COMPUTER

PREISSENKUNG:

21420 SCRIBE-Drucker mit Kabel	248.00

SCHNEIDER COMPUTER

22100 JOYCE PCW 8256	1799.00

IBM-PC-KOMPATIBLE

23001 COMMODORE

PREISSENKUNG:

23012 PC 10-II 640 K, 2 LW	1875.00

PREISSENKUNG:

23020 PC 10-II 2 LW, 20 MB	2675.00
23050 PC AT 40/20	3875.00
23055 PC AT 40/40 incl.40 MB HD	6498.00

23201 ZENITH

23240 Z 148 College PC o.Monit.	1845.00
23245 Z 148 Hercules kompatibel	2095.00
23260 Z 148 mit 20 MB o.Monitor	2499.00
23262 Z 148 HR komp.20 MB o.Mon	2895.00
23270 Z 148 mit 30 MB o.Monitor	2799.00
23272 Z 148 HR komp.30 MB o.Mon	3099.00
23290 ZENITH-Monitor ZVM gruen	229.00
23291 ZENITH-Monitor ZVM amber	229.00
23292 ZENITH-Monitor TTL	279.00

23301 SCHNEIDER

23310 PC 1512 SD 1 Laufwerk	1499.00
23321 PC 1512 DD, 2 LW, o. Mon.	1478.00
23351 PC 1512 SD+20 MB, o. Mon.	1999.00
23361 PC 1512 SD+30 MB, o. Mon.	2600.00
23380 Schneider SW-Monitor	299.00
23385 Schneider Farbmonitor	799.00
23387 SW-Mon.mit hrk.komp.Karte	685.00
23390 Plattenhalterung+Luefter	95.00

AGA Gr[illegible]
Smart [illegible]ch

PREISSENKUNG:

23855 20 MB incl.C[illegible]	[illegible]

PREISSENKUNG:

23856 20 MB incl.Co[illegible]	48.00
23862 30 MB-Festpla[illegible]	48.00
23872 HD-Controller [illegible]	[illegible]
23873 HD-Co roller 3[illegible]	[illegible]
23875 Parallel Druckerkabel	39.00
23876 Multi Printer Kabel	175.00
23880 Seriell Druckerkabel	49.00
23920 MICROSOFT WINDOWS	299.00

SINCLAIR COMPUTER

PREISSENKUNG:

28012 Spectrum 48 K	179.00

PREISSENKUNG:

28013 Spectrum Plus	225.00
28018 Sinclair QL Deutsche Ver.	399.00

PREISSENKUNG:

28022 3.5 Disk 720 K 1.Laufwerk	349.00

PREISSENKUNG:

28023 3.5 Disk 720 K 2.Laufwerk	349.00

PREISSENKUNG:

28024 Disk Controller fuer QL	249.00
28050 SEIK.GP50S anschlussfert.	175.00
28450 Parallelinterf. SPECTRUM	95.00
28460 Disk-Controll.f.SPECTRUM	248.00

TRIUMPH-ADLER

PREISSENKUNG:

33100 TA-PC 64 K RAM	169.00

ATARI COMPUTER

PREISSENKUNG:

34061 Atari 520 ST/M deutsch	598.00
34062 Atari Monitor SM 124	475.00
34064 Atari Floppy SF 354 0.5MB	274.00
34066 Atari Floppy SF 314 1 MB	568.00
34100 Atari 130 XE	265.00
34110 Atari 800 XE (64 K)	199.00

PREISSENKUNG:

34150 Atari 800 XL	149.00
34200 Atari 1050 Floppy 127 K	349.00
34220 Atari Recorder CX	79.00

ATARI 800 XL incl. Floppy 1050 komplett nur 399,-

ATARI 800 XL incl. Recorder CX 12 komplett nur 199,-

34315 1020 4-farb Druck/Plotter	199.00
34320 Atari 1029 Matrix-Drucker	299.00
34420 ST-FORTH f. 260/520 ST	98.00
34800 Software Pack f.800,130xe	10.00

DRUCKER

42230 SP 180 VC	465.00
42235 SP 180 AI	465.00
42250 SP 1000 VC Wie 800 an C64	499.00
42255 SP 1000 I IBM kompatibel	599.00
42260 SP 1000 AS Seriell	599.00
42267 SL 80 AI 24-Nadel-Drucker	875.00
42400 CENTRONICS GLP	365.00
42450 SEIKOSHA GP 100 AS	199.00

ISSEI.

6[illegible] SEIKO 6[illegible] A Farb.	299.00
00 ST/[illegible] 10 F. IBM	675.00
2505 STAR [illegible] 10 CENTRONICS	675.00
510 STAR NL 10 F. VC	675.00
2600 NEC P 6	1198.00
2610 NEC P 6 C(olor)	1575.00
2620 NEC P 7	1585.00
42630 NEC P 7 C(olor)	2395.00
42680 P 6 Cut-Sheet-Guide	32.00
42685 P 6 Unidir. Traktor	129.00
42690 P 7 Bidir. Traktor	399.00
42692 P 7 Cut-Sheet-Guide	39.00

TYPENRADDRUCKER

PREISSENKUNG:

43020 OLIVETTI DY 800 80 Z/Sek.	1998.00
43050 ASF 253 3-fach Formeinzug	1498.00
43100 Microscan MS 15 Cent-V24	575.00

DATENFERNUEBERTRAGUNG

44025 CDI Hitrans 300 C FTZ-Nr.	169.00

MONITORE

PREISSENKUNG:

47010 FERGUSON Monitor gruen	99.00
47013 FERGUSON Monitor amber	119.00
47016 Kabel FERGUSON-VC	5.00
47017 Kabel FERGUSON-Cinch	5.00
47018 Kabel FERGUSON-QL	5.00
47050 HIGHSCREEN Monitor Tuner	199.00
47060 KP 548 RGBI Stereo-Farbm.	795.00

PREISSENKUNG:

47070 Highscreen DP 602 - P 31	159.00

PREISSENKUNG:

47080 Highscreen DP 622 - LA	179.00

PREISSENKUNG:

47090 Highscreen DP 612 - P 39	199.00

PREISSENKUNG:

47092 Highscreen DP 652 - Schw.	179.00

SINCLAIR QL 128 K + HIGHSCREEN DP 652- Monitor nur 475,-

PREISSENKUNG:

47096 Highscreen DP 613 - TTL	149.00

PREISSENKUNG:

47105 Highscreen KP 501 RGBI	475.00

PREISSENKUNG:

47130 Thomson 36512 RGBI Mon.	635.00

MULTILIFE DISKETTEN

49001 5.25, 48 TPI contrast	9.95
49010 5.25, 1D1S brauner Karton	14.00
49011 5.25, 2D1S blauer Karton	19.00
49012 5.25, 1D2S schwarz.Karton	19.00
49013 5.25, 1D2S Wendedisketten	22.00
49014 5.25, 2D2S Nachfuellpack.	35.00
49016 5.25, 2D2S Plastikbox	39.00
49032 5 Stk. 3.5 Zoll Disk. 1DD	17.00
49042 5 Stk. 3.5 Zoll Disk. 2DD	22.00
49050 3.5 Zoll SONY Disk. 5 Stk	19.95

JOYSTICKS

49815 Quickshot I	9.95
49820 Quickshot II	15.00
49821 Quickshot II+	19.00
49825 Competition Pro-Joystick	29.00
49850 Joystick F.C16/PLUS4...	19.00

Öffnungszeiten: 10-18 Uhr, samstags von 10-13 Uhr

Ab 1.8. KIEL, Sophienblatt 74-78

kompetent + preiswert

WICHTIG! Herstellerbedingte Lieferzeiten! Aufgrund erhöhter Nachfrage ist nicht immer alles sofort lieferbar!

VOBIS

Deutschlands umsatzgrößter Microcomputer-Spezialist

VERSAND-ZENTRALE:
Postfach 1778
Rotter Bruch 32-34
5100 AACHEN
☎ 0241/500081
Tx 832389vobis d

FILIALEN:

AACHEN
Viktorstr. 74 · 0241/54 31 00

Seit 4.7. AUGSBURG
Jakoberstr. 16 · 0821/152349

BERLIN 30
Kurfürstenstr. 101 · 030/2 13 94 80

BIELEFELD
Herforder Str. 106 · 0521/6 38 78

BREMEN
Violenstraße 37 · 0421/32 04 20

DORTMUND
Hamburger Str. 190 · 0231/57 30 72

DÜSSELDORF
Wielandstr. 21 · 0211/35 99 64

Seit 6.6 ESSEN
Huyssenallee 3 · 0201/23 17 74

FRANKFURT
Frankenallee 207/209 · 069/73 40 49

HAMBURG
Krohnstamp 15 · 040/2 79 46 76

HANNOVER
Berliner Allee 47 · 0511/81 65 71

KARLSRUHE
Kriegsstr. 27/29 (am BGH) 0721/37 82 68

KONSTANZ
Kreuzlinger Str. 18 · 07531/1 55 80

KÖLN
Mathiasstr. 24-26 · 0221/24 86 42

MÜNCHEN
Aberlestr. 3 · 089/77 21 10

NÜRNBERG
Vordere Ledergasse 8 · 0911/23 29 95

STUTTGART
Marienstr. 11-13 · 0711/60 63 36

Machen Sie von unserem bequemen Teilzahlungsangebot Gebrauch! Mit dem

CK-Kauf
VOBIS COMPUTER-KREDIT

Kaufen Sie jetzt, zahlen Sie später!

Fordern Sie noch heute den Kurz-Kredit-Bestellschein an!